PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 11/14	A1	(11) International Publication Number: WO 95/13580 (43) International Publication Date: 18 May 1995 (18.05.95)

(21) International Application Number: PCT/US94/12915

(22) International Filing Date: 9 November 1994 (09.11.94)

(30) Priority Data:
150,488 9 November 1993 (09.11.93) US

(71) Applicant: ARCADA SOFTWARE [US/US]; Suite 1101, 37 Skyline Drive, Lake Mary, FL 32746 (US).

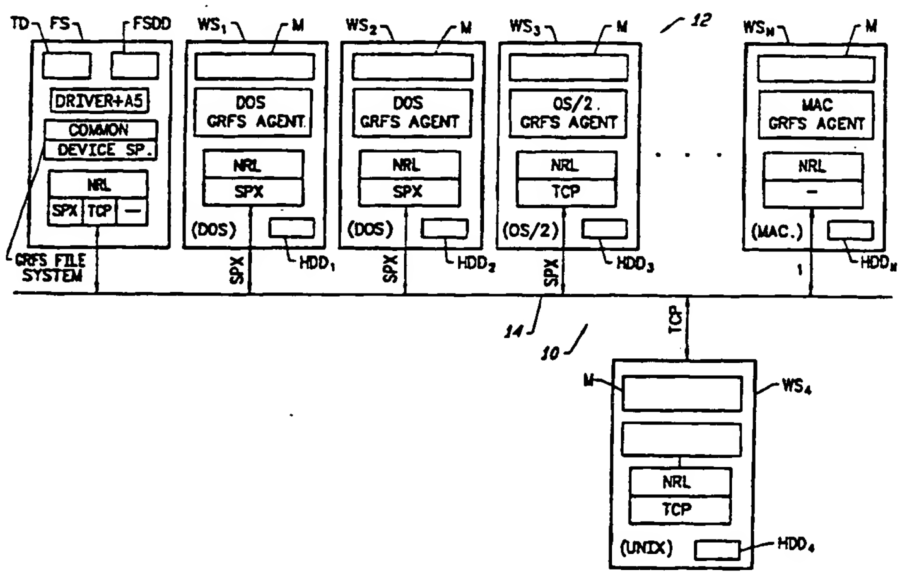
(72) Inventors: FLETCHER, Douglas, J.; 340 Wekiva Trail West, Longwood, FL 32779 (US). DEVOS, Steven, Robert; 1529 3rd Street, Kirkland, WA 98033 (US).

(74) Agent: FLIESLER, Martin, C.; Fliesler, Dubb, Meyer and Lovejoy, Suite 400, Four Embarcadero Center, San Francisco, CA 94111-4156 (US).

(81) Designated States: CA, CN, JP, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: DATA BACKUP AND RESTORE SYSTEM FOR A COMPUTER NETWORK



(57) Abstract

A computer network having a number of workstations running disparate operating systems and a file server having a tape driver for backup and restore of data on the network. The filter server runs a generic remote file system (GRFS) and workstations run GRFS agent programs which allow the GRFS file system to access data within a workstation having a given GRFS agent program. The GRFS file system interfaces with each GRFS agent program via a command/response paradigm, with the messages being structured to support the disparate operating systems for backup and restore, to allow data to be interchanged between the disparate operating systems, and to allow independent multiple users of the network to request simultaneously backup or restore.

data area is at most 1,024 bytes. Furthermore, there are several fields within the DBLK structure, which are actually pointers to information within the DBLK data area. These pointers are generated as offsets from the beginning of the DBLK structure. For example, if the DBLK common area is 80 bytes long and the first item within the data area is the object's name, then the object name field would be set to 80 in order to point to the first byte following the DBLK common structure. The individual fields within the common DBLK structure that are manipulated by the GRFS agent programs are described in detail below under the heading "DBLK Fields".

In order to implement a backup and restore function for a given computer 12, that computer 12 should advertise its capability for this purpose. Not every computer 12 in the network 10 is necessarily running a GRFS agent program so as to be able to have its data backed up. Consequently, the GRFS agent programs will "advertise" their capability as a GRFS agent over the network 10. This may be accomplished using the NRL resource advertisement function. The GRFS agent resource advertisement publishes the logical name of the particular agent's root DLE, as well as various flags which are used by the GRFS file system to control access to the GRFS agent. The format of the GRFS agent advertisement structure is as follows:

```
struct grfs_ws_adver_struct
      {
      CHAR major_ver;
      CHAR minor_ver;
      CHAR agent_type;
      CHAR flags;
      CHAR name[MAX_WORKSTATION_NAME_LEN];
      }
```

GRFS agents use character representations of the values in the version and flags fields. For example, the major.minor version of a particular GRFS agent might be

DATA BACKUP AND RESTORE SYSTEM FOR A COMPUTER NETWORK

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to a system for protecting data through backup and restore operations, and more particularly to backup and restore software for protecting data which is processed on a computer network

Description of the Related Art

In order to ensure that original data stored on a medium such as a disk is not lost or damaged, a copy of that data is stored on another medium. Should the original data be lost or damaged, then the copy may be accessed to reproduce the original data. This process of copying and reproducing is generally known as backup and restore. Typically, original data are stored on a hard or floppy disk of a computer disk drive and are backed up to and restored from tape media of a tape drive.

Backup and restore of the data are simple in a system that has a single standalone computer, having a given operating system and one or more disk drives, that interfaces with a tape drive system. A relatively simple backup and restore program can be used that interfaces with the computer operating system to backup data including files and directories stored on a hard disk to the tape drive and to restore such data from the tape drive onto the hard disk.

Computer networks have evolved and this has placed greater demands on backup and restore systems. A computer network may include a number of computers each with its own hard and/or floppy disk drive, all of which are networked together on a common bus. For example, the computers on the network may include one or more

"0043ONE_WOLF"

major version = 0
minor version = 0
Unix agent
user name required
password required
DLE name = "ONE_WOLF"

Fig. 4 illustrates a sequence of GRFS command and response messages in simplified form to backup data on the tape drive TD of the file server FS. This Fig. 4 gives the example of backing up a 5000 byte file named COMMAND.COM which is stored on a "DRIVEC" of a given workstation named "DougCompaq". It is assumed that the given workstation WS has advertised over the network 10 sufficient information so that the GRFS file system can create the first command message shown in Fig. 4 as ATTACH_DLE(.

To begin the 5000 byte backup, the workstation user will, via a given user interface 18, cause a display on a monitor M of devices and subdevices. The user will then select a given subdevice (e.g., DRIVEC in the example of Fig. 4), resulting in the user interface displaying on monitor M names of various files and directories. The user will then select the file name to be backed up (COMMAND.COM in the example) resulting in the submission of a tape backup job for the file server FS in the network 10.

Next, the sequence of GRFS file system command messages and GRFS agent response messages will occur as shown in order in the simplified Fig. 4. The sequence, as illustrated, commences with the GRFS command message ATTACH_DLE(naming "DougCompaq" (dle.id=01) and completes with the final GRFS agent response message DETACH_DLE_STAT() by which DougCompaq (dle.id=01) is detached. Thus, the file COMMAND.COM will be read from DRIVEC and written onto the tape drive TD of the file server FS for network 10.

obviously increases as more and more disparate operating systems are added to the network via the computers on which they run.

In general, prior backup and restore systems for computer networks are limited to the number of different types of operating systems that can be supported. This places expansion limitations on the network in terms of adding computers running additional types of operating systems. Also, these backup and restore systems do not have the capability of interchanging data between different operating systems. Furthermore, bottlenecks occur and productivity is limited with prior backup and restore operations since multiple users cannot simultaneously request these operations.

SUMMARY OF THE INVENTION

The present invention provides a backup and restore system for use on a computer network having computers running disparate operating systems. Backup and restore software has modules including a backup engine containing, among other components, a generic remote file system (GRFS file system) and GRFS agents, being loadable on a computer network having a plurality of computers including, for example, at least one file server and at least one workstation. The GRFS file system may run on one computer, e.g., the file server of the network, and each GRFS agent may run on another computer, e.g., a workstation, on the network. The GRFS file system running on the one computer, i.e., the file server in this example, is allowed to access a file system of the other computer via the GRFS agent on that other computer to backup and restore data on that computer.

The GRFS file system and each GRFS agent interface with one another over the computer network by a set of defined messages. This messaging system is based on a

is used by the backup application's tape format module and is written to the backup media of tape device TD. A well-known Microsoft Tape Format Version 1.0 Specification describes stream header structures and also contains a list of pre-defined stream header id values. The size field must be set to the number of bytes contained in the succeeding data stream and should only be set in the first stream header structure for a particular data stream, i.e., if the stream header id value is 0, then the size field does not need to be set.

An example is presented below of what a Macintosh GRFS agent would return in the GRFS_READ_OBJ_STAT messages when a file with a 2000 byte resource fork and a 4000 byte data fork is being backed up. This example also assumes that a GRFS data buffer limit is 1000 bytes.

strm_header.id=STRM_MAC_RESOURCE strm_header.size=2000	(returns 1st 1000 bytes of resource fork)
strm_header.id=STREAM_INVALID strm_header.size=0	(returns last 1000 bytes of resource fork)
strm_header.id=STRM_NORMAL_DATA strm_header.size=2000	(returns 1st 1000 bytes of data fork)
strm_header.id=STREAM_INVALID strm_header.size=0	(returns next 1000 bytes of data fork)
strm_header.id=STREAM_INVALID strm_header.size=0	(returns next 1000 bytes of data fork)
strm_header.id=STREAM_INVALID strm_header.size=0	(returns last 1000 bytes of data fork)

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Fig. 1 illustrates one example of a computer network 10 which stores, manipulates, and otherwise processes data. The network 10 has a number of computers 12 which can communicate with one another over a network bus 14. In the example of Fig. 1, the computers 12 include a file server FS and a plurality of workstations WS_1, WS_2, WS_3, WS_4,...WS_n. Each of the workstations WS_1-WS_n has a display monitor M and the workstations WS_1-WS_n include hard disk drives HDD_1-HDD_n. The file server FS has its own large file server disk drive FSDD and a tape drive TD upon which to backup to and restore from data on the network 10.

Every workstation WS_1-WS_n may be running the same operating system OS, or each workstation WS_1 through WS_n may be running a disparate operating system, or there may be disparate groups of workstations with each group running the same operating system. For example, workstation WS_1 and workstation WS_2 may both be running the operating system known as DOS, workstation WS_3 may be running the operating system known as OS/2, workstation WS_4 may be running the operating system known as UNIX, workstation WS_n may be running the operating system known as Macintosh, and other workstations, not shown, or which may be added to the network 10, may run the operating system known as Windows. Furthermore, the computers 12 in the network 10 may be utilizing user interfaces such as those known as the DOS user interface, Windows graphical user interface, and a server-based NLM (NetWare Loadable Module) interface.

The computer network 10 may be, for example, running the operating system software known as NetWare 3.X or 4.X which is produced by Novell, Inc., of Provo, Utah. NetWare is designed to manage programs and data among the several computers 12 of the network 10. Fig. 1 also

alignment. The GRFS messages are defined with a "least common denominator" alignment that would apply to the above-noted major operating systems. Thus, for example, a given network 10 which may include workstations running only DOS, OS/2, and Macintosh, may be expanded to include a workstations running UNIX and/or Windows. In other words, the present invention supports a scalable network for backup and restore purposes from a small or departmental local area network (LAN) to a large or enterprise wide area network (WAN).

Furthermore, the message structure enables multiple users working at multiple computers 12 on the network 10 to request simultaneously backup and restore of objects. This structure enables the GRFS file system to create a unique request id for every GRFS command message. Consequently, the GRFS file system can communicate simultaneously with multiple GRFS agents and, therefore, multiple users of the network 10 who at the same time want to have backup and/or restore operations performed. The present invention will manage these requests such that they are placed in a job queue in the file server FS, thereby allowing each user to operate independently from any other user on the network 10 and without waiting access to the backup and restore system.

While each user can independently manage his/her own data on a given workstation, backup and restore of data on the entire network 10 can be centrally managed at a single location by, for example, a network administrator, from a given workstation or file server, or a system console.

The remaining portion of this specification describes in much more detail the structure of the command/response messages, followed by a detailed description of the individual fields of the GRFS common

having the corresponding operating system in order to access the file system of that given computer. Thus, for example, the DOS GRFS agent 20 will run on a DOS workstation WS_1, the OS/2 GRFS agent will run on the OS/2 workstation WS_2, etc. The package 16 also has a backup engine 22 running on the file server FS and includes a tape controller device driver and tape positioner to control the mechanical operation of the tape drive TD, a common file system, and at least one device specific file system. The latter is a GRFS file system which interfaces with GRFS agents 20 via messages described in more detail below.

Fig. 3 illustrates the network 10, but modified to include the software 16. As shown, the backup engine 22 is installed at the file server FS, while the DOS, OS/2, UNIX, and Macintosh GRFS agents are installed on the respective workstations WS_1-WS_2, WS_3, WS_4, and WS_n. In this example, the computer network 10 does not have a computer 12 running a Windows operating system. Should the network 10 be expanded to include a Windows workstation, then the Windows GRFS agent of the software 16 would be installed at that workstation. While not specifically illustrated, a workstation user also can opt to have installed one of the user interfaces 18 for tape backup and restore purposes, that is the same as that already on a workstation for other purposes.

As indicated above, a GRFS agent is a program which runs on a network computer such as the given workstation WS, and which allows the GRFS file system running on another computer, such as the file server FS, to access the file system within the given GRFS agent's computer. This access is accomplished by use of an interface between the GRFS file system and the given GRFS agent over the network bus 14. Specifically, the interface is defined by a set of GRFS messages which are documented in

SPECIFIC DESCRIPTION OF COMMAND/RESPONSE MESSAGES

3.0 Using GRFS Command and Response Messages

This following sections provide the information necessary to implement each of the GRFS command and response messages.

3.1 GRFS_ATTACH_DLE_, GRFS_ATTACH_DLE_STAT

After establishing an NRL session with the GRFS agent, the first GRFS command the backup application will send to the GRFS agent is the GRFS_ATTACH_DLE command. The GRFS_ATTACH_DLE command message contains the following parameters:

dle_name: This field contains the name of the DLE that the backup application desires to attach to. The dle_name field is encrypted in conjunction with the encryption done on the password field. The encryption/decryption method used by GRFS is described in the GRFS encryption section of this document.

bec_flags: This field contains a bit-mapped value which defines the configuration options chosen by the backup application program. The values defined for use in this field are as follows:

BEC_BACKUP_FILES_INUSE 0x01

If this flag is set, then the GRFS agent should attempt to open files even if they are already in use by another process.

BEC_EXTENDED_DATE_SUPPORT 0x02

If this flag is set, then the backup application knows how to handle the ACCESS DATE and ARCHIVE DATE fields in the GRFS DBLK, so if the agent's OS platform supports these time-stamps, they should be provided in DBLKs.

BEC_SET_ARCHIVE_FLAG 0x04

If this flag is set and the agent's OS platform supports an object "ARCHIVED" flag, then the GRFS agent should set an object's ARCHIVED flag after the object is closed during the backup operation.

BEC_RESTORE_SECURITY 0x08

If this flag is set and the agent's OS platform has support for security specific data forks (ie ACL support for LANMAN OS/2), then security information should be restored during the restore operation.

BEC_GET_HIDDEN_FILES 0x10

This flag controls whether "hidden" objects should be returned while processing GRFS_FIND_FIRST_OBJ and GRFS_FIND_NEXT_OBJ commands.

retcode:	This UINT16 field is used by GRFS status messages to hold the return code of the GRFS command.
request_id:	This UINT32 field contains a value which is generated by the GRFS file system for GRFS command messages and must be returned unchanged in the corresponding GRFS response message.

In the detailed description of the specific GRFS messages below under the heading "Specific Description of Command/Response Messages", the number of parameters associated with a given GRFS agent is assumed not to include the above GRFS common message header. The messages use two major structures to define GRFS objects. These two major GRFS object types are a drive list element (DLE) objects, which are logical devices, and file system objects, which are files and directories. The GRFS messages use DLE structures to reference drive list element objects and DBLK (descriptor block) structures to reference file system objects.

A DLE is a structure that contains information about individual data storage devices which can be accessed for backup and restore. The DLE structure contains the following types of information: logical device name, access password, file system delimiter, etc.

A DLE structure also supports a hierarchical structure. A DLE can be a "parent" DLE and can have "children" DLEs associated with it. For example, this is the case for a Novell server file system. For a Novell server, a DLE structure is created which is associated with the server and then DLEs for each volume on the server are created. The same situation can occur with a GRFS agent should that agent advertise or publish on the network 10 the workstation name as a DLE and then use children DLEs to advertise individual areas which can be accessed as logical units.

special_word: This field is not used.

max_obj_bsize: This field contains the size of the buffer that the GRFS file system would like to use when transferring object data to/from the GRFS agent. This buffer size is the size of the object data buffer, not the size of the GRFS message buffer. GRFS message buffers are larger than the object data buffer size because the GRFS message buffer must include the 8-byte common header as well as the miscellaneous parameters (obj_id, stream_info, etc) used by the GRFS_WRITE_OBJ, GRFS_VERIFY_OBJ, and GRFS_READ_OBJ_STAT messages.

The GRFS object buffer size is a negotiated size, so if the value contained in the max_obj_bsize is larger than the agent would like, the agent can return a smaller value in the GRFS_ATTACH_DLE_STAT max_obj_bsize field. The GRFS file system will use the value returned by the GRFS agent if it is smaller than the default file system object data buffer size.

dle_parent: This field contains the DLE handle for the parent of the DLE being attached to if a parent DLE exists. If a parent DLE does not exist, then this field is set to 0.

cmpr_type: This field is not currently supported.

user_name: This field contains the user name supplied by the backup application. This field will be filled only if the DLE is defined as requiring a user name.

password: This field contains the password supplied by backup application if the DLE is defined as requiring a password. Even if the DLE requires no password, this field will appear to have a value until it is decrypted. Please see the section on DLE name/Password decryption for more information.

The proper response message for a GRFS_ATTACH_DLE is the GRFS_ATTACH_DLE_STAT message. The parameters associated with the GRFS_DLE_ATTACH_STAT message are described below.

dle_id: This field must be set to the DLE id which the GRFS agent wishes to use to identify the DLE. The DLE id is a 32-bit value which the backup application will use in future GRFS commands to identify the DLE to be operated upon. Typically, the GRFS agent will create DLE ids as a pointer to a structure of an index into an array. The DLE id can be any value except 0.

max_connects: This field should be set to the maximum number of concurrent GRFS sessions which the agent is capable of.

data area is at most 1,024 bytes. Furthermore, there are several fields within the DBLK structure, which are actually pointers to information within the DBLK data area. These pointers are generated as offsets from the beginning of the DBLK structure. For example, if the DBLK common area is 80 bytes long and the first item within the data area is the object's name, then the object name field would be set to 80 in order to point to the first byte following the DBLK common structure. The individual fields within the common DBLK structure that are manipulated by the GRFS agent programs are described in detail below under the heading "DBLK Fields".

In order to implement a backup and restore function for a given computer 12, that computer 12 should advertise its capability for this purpose. Not every computer 12 in the network 10 is necessarily running a GRFS agent program so as to be able to have its data backed up. Consequently, the GRFS agent programs will "advertise" their capability as a GRFS agent over the network 10. This may be accomplished using the NRL resource advertisement function. The GRFS agent resource advertisement publishes the logical name of the particular agent's root DLE, as well as various flags which are used by the GRFS file system to control access to the GRFS agent. The format of the GRFS agent advertisement structure is as follows:

```
struct grfs_ws_adver_struct
    {
    CHAR major_ver;
    CHAR minor_ver;
    CHAR agent_type;
    CHAR flags;
    CHAR name[MAX_WORKSTATION_NAME_LEN];
    }
```

GRFS agents use character representations of the values in the version and flags fields. For example, the major.minor version of a particular GRFS agent might be

1.3, so that agent would advertise the version numbers as "1" and "3", respectively.

The GRFS major version number is used to control which GRFS agents can be accessed by the GRFS file system. The GRFS major version number of the GRFS file system and the GRFS agent must match exactly or no information of the existence of that GRFS agent will be given. The GRFS minor version number may be used for informational purposes only.

The agent_type field is used to define the type of GRFS agent. For example, the following values may be defined for this field:

DOS	1
OS2	2
MACINTOSH	3
UNIX	4

The GRFS flags field is a bit-mapped value with the following flags currently defined:

GRFS_WS_PASSWORD_REQ	0x01
GRFS_WS_USER_REQ	0x02

Combining all the GRFS resource advertisement fields leads to the following examples of GRFS agent advertisements:

NRL Resource	Decoded As
"1211RATBOY_486"	major version = 1 minor version = 2 DOS agent no user name required password required DLE name = "RATBOY_486"
"1020SLEDGEHAMMER"	major version = 1 minor version = 0 OS/2 agent no user name required no password required DLE name = "SLEDGEHAMMER"

"0043ONE_WOLF"	major version = 0 minor version = 0 Unix agent user name required password required DLE name = "ONE_WOLF"

Fig. 4 illustrates a sequence of GRFS command and response messages in simplified form to backup data on the tape drive TD of the file server FS. This Fig. 4 gives the example of backing up a 5000 byte file named COMMAND.COM which is stored on a "DRIVEC" of a given workstation named "DougCompaq". It is assumed that the given workstation WS has advertised over the network 10 sufficient information so that the GRFS file system can create the first command message shown in Fig. 4 as ATTACH_DLE(.

To begin the 5000 byte backup, the workstation user will, via a given user interface 18, cause a display on a monitor M of devices and subdevices. The user will then select a given subdevice (e.g., DRIVEC in the example of Fig. 4), resulting in the user interface displaying on monitor M names of various files and directories. The user will then select the file name to be backed up (COMMAND.COM in the example) resulting in the submission of a tape backup job for the file server FS in the network 10.

Next, the sequence of GRFS file system command messages and GRFS agent response messages will occur as shown in order in the simplified Fig. 4. The sequence, as illustrated, commences with the GRFS command message ATTACH_DLE(naming "DougCompaq" (dle.id=01) and completes with the final GRFS agent response message DETACH_DLE_STAT() by which DougCompaq (dle.id=01) is detached. Thus, the file COMMAND.COM will be read from DRIVEC and written onto the tape drive TD of the file server FS for network 10.

Fig. 5 shows a sequence of GRFS command/response messages to restore information backed up on the file server FS of the network 10. In this example, it is assumed that a 5000 byte file named CONFIG.SYS has been backed up from a given workstation and is to be restored to DRIVEC of the workstation DougCompaq. After the workstation user has selected the file CONFIG.SYS using the user interface to select the file CONFIG.SYS for restore, the sequence of GRFS command/response messages will proceed as shown in Fig. 5. The sequence begins with the GRFS command message ATTACH_DLE(and completes with the GRFS response message DETACH_DLE_STAT(). The file CONFIG.SYS will be read from the tape drive TD and restored onto DRIVEC.

As mentioned previously, the command/response messages are structured such that objects such as files and directories may be backed up from a GRFS agent running one operating system, e.g., OS/2, and restored to a GRFS agent running another operating system, e.g., DOS. This is accomplished by the messages containing a structure GRFS_STREAM_INFO. This structure has the following definition:

```
struct GRFS_STREAM_INFO {
    UNET32        id;
    UNET16        fs_attrib;
    UNET16        tf_attrib;
    UNET64        size;
    }
```

When the backup application is reading an object, the GRFS_READ_OBJ_STAT response message contains a GRFS_STREAM_INFO structure. The GRFS agent program must set the id field of the first GRFS_READ_OBJ_STAT response message of each individual data stream to the appropriate value for the agent's particular operating system. Succeeding GRFS_READ_OBJ_STAT messages for the stream must have the stream header id field set to 0 (STREAM_INVALID). The data in the stream info structure

is used by the backup application's tape format module and is written to the backup media of tape device TD. A well-known Microsoft Tape Format Version 1.0 Specification describes stream header structures and also contains a list of pre-defined stream header id values. The size field must be set to the number of bytes contained in the succeeding data stream and should only be set in the first stream header structure for a particular data stream, i.e., if the stream header id value is 0, then the size field does not need to be set.

An example is presented below of what a Macintosh GRFS agent would return in the GRFS_READ_OBJ_STAT messages when a file with a 2000 byte resource fork and a 4000 byte data fork is being backed up. This example also assumes that a GRFS data buffer limit is 1000 bytes.

strm_header.id=STRM_MAC_RESOURCE strm_header.size=2000	(returns 1st 1000 bytes of resource fork)
strm_header.id=STREAM_INVALID strm_header.size=0	(returns last 1000 bytes of resource fork)
strm_header.id=STRM_NORMAL_DATA strm_header.size=2000	(returns 1st 1000 bytes of data fork)
strm_header.id=STREAM_INVALID strm_header.size=0	(returns next 1000 bytes of data fork)
strm_header.id=STREAM_INVALID strm_header.size=0	(returns next 1000 bytes of data fork)
strm_header.id=STREAM_INVALID strm_header.size=0	(returns last 1000 bytes of data fork)

When the backup application is restoring an object, the GRFS commands (GRFS_WRITE_OBJ, GRFS_VERIFY_OBJ) will also contain a GRFS_STREAM_INFO structure. The GRFS agent must examine the stream header id value to determine whether the data stream type is supported on the agent's operating system platform. If the data stream type is not supported the GRFS agent should set the response message retcode to FS_DONT_WANT_STREAM. This will cause the backup application to skip to the next data stream or the next object if at the last data stream for a particular object. For instance, if an object was backed up from an OS/2 agent which supports a normal data stream, an extended attribute (EA) data stream, and an access control list (ACL) data stream, then if the object is restored to a DOS agent, the DOS agent will return FS_DONT_WANT_STREAM when it receives GRFS_WRITE_OBJ commands with stream header id values that indicate either EA or ACL data streams are being restored since this data is not supported by DOS. The DOS agent will accept the normal data stream which it does support. Thus, this functionality allows objects to be backed up from an agent running on one operating system and restored to an agent running on another operating system.

As also mentioned above, the message structure is defined as well, such that backup and restore can be supported with respect to most operating systems, including the current major operating systems which are DOS, OS/2, Macintosh, Windows, and UNIX. Each operating system will have its own data structures aligned differently from one another. For example, one operating system may have a 1-byte alignment where a data byte may be placed anywhere, whereas another operating system may have a 2-byte alignment where a data byte may be placed in either an even or odd byte location. Other operating systems, for example, may have what is known as a 4-byte

alignment. The GRFS messages are defined with a "least common denominator" alignment that would apply to the above-noted major operating systems. Thus, for example, a given network 10 which may include workstations running only DOS, OS/2, and Macintosh, may be expanded to include a workstations running UNIX and/or Windows. In other words, the present invention supports a scalable network for backup and restore purposes from a small or departmental local area network (LAN) to a large or enterprise wide area network (WAN).

Furthermore, the message structure enables multiple users working at multiple computers 12 on the network 10 to request simultaneously backup and restore of objects. This structure enables the GRFS file system to create a unique request id for every GRFS command message. Consequently, the GRFS file system can communicate simultaneously with multiple GRFS agents and, therefore, multiple users of the network 10 who at the same time want to have backup and/or restore operations performed. The present invention will manage these requests such that they are placed in a job queue in the file server FS, thereby allowing each user to operate independently from any other user on the network 10 and without waiting access to the backup and restore system.

While each user can independently manage his/her own data on a given workstation, backup and restore of data on the entire network 10 can be centrally managed at a single location by, for example, a network administrator, from a given workstation or file server, or a system console.

The remaining portion of this specification describes in much more detail the structure of the command/response messages, followed by a detailed description of the individual fields of the GRFS common

DBLK structure which may be manipulated by GRFS agent programs.

SPECIFIC DESCRIPTION OF COMMAND/RESPONSE MESSAGES

3.0 Using GRFS Command and Response Messages

This following sections provide the information necessary to implement each of the GRFS command and response messages.

3.1 GRFS_ATTACH_DLE_, GRFS_ATTACH_DLE_STAT

After establishing an NRL session with the GRFS agent, the first GRFS command the backup application will send to the GRFS agent is the GRFS_ATTACH_DLE command. The GRFS_ATTACH_DLE command message contains the following parameters:

dle_name: This field contains the name of the DLE that the backup application desires to attach to. The dle_name field is encrypted in conjunction with the encryption done on the password field. The encryption/decryption method used by GRFS is described in the GRFS encryption section of this document.

bec_flags: This field contains a bit-mapped value which defines the configuration options chosen by the backup application program. The values defined for use in this field are as follows:

BEC_BACKUP_FILES_INUSE 0x01

If this flag is set, then the GRFS agent should attempt to open files even if they are already in use by another process.

BEC_EXTENDED_DATE_SUPPORT 0x02

If this flag is set, then the backup application knows how to handle the ACCESS DATE and ARCHIVE DATE fields in the GRFS DBLK, so if the agent's OS platform supports these time-stamps, they should be provided in DBLKs.

BEC_SET_ARCHIVE_FLAG 0x04

If this flag is set and the agent's OS platform supports an object "ARCHIVED" flag, then the GRFS agent should set an object's ARCHIVED flag after the object is closed during the backup operation.

BEC_RESTORE_SECURITY 0x08

If this flag is set and the agent's OS platform has support for security specific data forks (ie ACL support for LANMAN OS/2), then security information should be restored during the restore operation.

BEC_GET_HIDDEN_FILES 0x10

This flag controls whether "hidden" objects should be returned while processing GRFS_FIND_FIRST_OBJ and GRFS_FIND_NEXT_OBJ commands.

BEC_GET_SYSTEM_FILES 0x20

This flag controls whether "system" objects should be returned while processing GRFS_FIND_FIRST_OBJ and GRFS_FIND_NEXT_OBJ commands.

BEC_PROC_EMPTY_DIRS 0x40

This flag controls whether directories which are empty should be returned while processing GRFS_FIND_FIRST_OBJ and GRFS_FIND_NEXT_OBJ commands.

special_word: This field is not used.

max_obj_bsize: This field contains the size of the buffer that the GRFS file system would like to use when transferring object data to/from the GRFS agent. This buffer size is the size of the object data buffer, not the size of the GRFS message buffer. GRFS message buffers are larger than the object data buffer size because the GRFS message buffer must include the 8-byte common header as well as the miscellaneous parameters (obj_id, stream_info, etc) used by the GRFS_WRITE_OBJ, GRFS_VERIFY_OBJ, and GRFS_READ_OBJ_STAT messages.

The GRFS object buffer size is a negotiated size, so if the value contained in the max_obj_bsize is larger than the agent would like, the agent can return a smaller value in the GRFS_ATTACH_DLE_STAT max_obj_bsize field. The GRFS file system will use the value returned by the GRFS agent if it is smaller than the default file system object data buffer size.

dle_parent: This field contains the DLE handle for the parent of the DLE being attached to if a parent DLE exists. If a parent DLE does not exist, then this field is set to 0.

cmpr_type: This field is not currently supported.

user_name: This field contains the user name supplied by the backup application. This field will be filled only if the DLE is defined as requiring a user name.

password: This field contains the password supplied by backup application if the DLE is defined as requiring a password. Even if the DLE requires no password, this field will appear to have a value until it is decrypted. Please see the section on DLE name/Password decryption for more information.

The proper response message for a GRFS_ATTACH_DLE is the GRFS_ATTACH_DLE_STAT message. The parameters associated with the GRFS_DLE_ATTACH_STAT message are described below.

dle_id: This field must be set to the DLE id which the GRFS agent wishes to use to identify the DLE. The DLE id is a 32-bit value which the backup application will use in future GRFS commands to identify the DLE to be operated upon. Typically, the GRFS agent will create DLE ids as a pointer to a structure of an index into an array. The DLE id can be any value except 0.

max_connects: This field should be set to the maximum number of concurrent GRFS sessions which the agent is capable of.

max_opens_per_connect: This field should be set to the maximum number of objects which can be opened simultaneously per GRFS session.

process-ddbs: This field is not currently supported.

max_obj_bsize: This field should be set to the maximum object data buffer size the agent wishes to use. The maximum GRFS message size is greater than the maximum object data buffer size because of the additional parameters in the GRFS messages which convey object data.

cmpr_type: This field is not currently supported.

supports_children: This field is a BOOLEAN flag which should be set to 0 if the DLE does not support children. A non-zero value declares the DLE as supporting children DLEs. A DLE declared as supporting children DLEs CANNOT support file system objects as well. Either a DLE supports children DLEs or file system objects. Never both.

path_len: This field should be set to the length of the string (including the '/0' terminator) returned in the current_path field. Current GRFS agent implementations will always start in the logical root directory of DLEs when they are attached, so the current_path field should always be set to ** and the path_len field set to 1.

current_path: This field should be set to the current path of the DLE being attached to. As described above, at DLE attachment time, the current path will be the logical root of the DLE, so the current path is empty (**).

3.2 GRFS_FIND_FIRST_DLE, GRFS_FIND_NEXT_DLE, GRFS_FIND_DLE_STAT

The GRFS_FIND_FIRST_DLE and GRFS_FIND_NEXT_DLE commands are used by the backup application program to enumerate children DLEs for DLEs which are declared as supporting children DLEs. The sole parameter associated with these two commands is the dle_id parameter. The backup application will supply the dle_id value which was previously returned by a GRFS_ATTACH_DLE_STAT response message. The GRFS agent should respond with a GRFS_FIND_DLE_STAT message to both the GRFS_FIND_FIRST_DLE and GRFS_FIND_NEXT_DLE command.

It is the responsibility of the GRFS agent to determine the sequence and keep track of the children DLEs as they are being enumerated. The parameter in the GRFS_FIND_DLE_STAT response message are described below.

dle_name: This field should contain the name of DLE which is being enumerated. The value must be a null-terminated string.

path_delim: This field should contain the ASCII code of the character used by the agent's file system.

passwd_req: This field is a boolean flag and should be set to 0 if no password is required to attach to the DLE. A non-zero value in this field indicates that a password is required.

user_req: This field is a boolean flag and should be set to 0 if no user name is required to attach to the DLE. A non-zero value in this field indicates that a user name is required in order to attach to the DLE.

dle_writable: This field is a boolean flag used to indicate whether restore operations are permitted on the DLE. Setting this value to 0 will prevent the backup application from attempting restore operations.

last_access_supported: This field is a boolean flag used to indicate whether the DLE's file system supports the last access date information. This field is used by the Backup application to determine whether file-grooming is supported for this device.

os_id:

os_ver:

fs_type:

cryp_type: This field is not currently used.

cmpr_type: This field is not currently used.

more_flag: This field is a boolean flag and should be used by GRFS agents to indicate that the DLE being returned is the last child DLE available. If the GRFS agent is incapable of knowing ahead of time whether this is the

last DLE, then this field can always be set to a non-zero value (TRUE). This will force the backup application to sent GRFS_FIND_NEXT_DLE commands until the GRFS agent responds with a FS_NO_MORE return code.

3.3 GRFS_DETACH_DLE, GRFS_DETACH_DLE_STAT

The GRFS_DETACH_DLE command is used by the backup application when it no longer needs to access a DLE. The message has only one command specific parameter, the dle_id of the DLE which the backup application wishes to detach from. DLEs will always be detached in the reverse order to which they were attached. In other words the last DLE which was attached to will be the first to be detached from. When a DLE is detached, the GRFS agent can free any resources associated with the attached DLE. The GRFS_DETACH_DLE_STAT message is the response type for the GRFS_DETACH_DLE command.

3.4
GRFS_FIND_FIRST_OBJ, GRFS_FIND_NEXT_OBJ, and GRFS_FIND_OBJ_STAT

The backup application uses the GRFS_FIND_FIRST_OBJ command to begin scanning for file system objects. GRFS agents must take into account the GRFS find object mask flags which were supplied in the GRFS_ATTACH_DLE command. These flags specify whether HIDDEN and SYSTEM objects should be returned for GRFS_FIND_FIRST_OBJ and GRFS_FIND_NEXT_OBJ commands. The parameters associated with find first command are explained below.

dle_id: This field contains the id of the DLE that the backup application wishes to scan.

find_type: This field contains one of these values:

0x00 -return all object types found
0x01 -return only directory objects found

sname: This field contains the search string qualifier. Normally this field will contain the string"*.*". The string "*.*" means that all objects that meet the find_type criteria should be returned.

3.4.1 GRFS Agent Path Generation

When a GRFS agent is creating the path string used for its file system's "FindFirst" system call, the following components must be included to create the correct path string. The path string must begin with the base directory of the DLE. The DLEs current path is then appended to the path string. Finally the sname parameter is appended to the path string. The GRFS agent must also supply path delimeters wherever required. An example of a "FindFirst" path string created by the OS/2 GRFS agent is presented below:

DLE base path: "C:\DOCS"

DLE current path: "GRFS\DESIGN"

sname: "*.*"

The GRFS agent creates the path string: "C:\DOCS\GRFS\DESIGN*.*"

Agents are responsible for keeping track of when path delimeters must be inserted. For example when OS/2 GRFS agent publishes the root directory of a disk drive, the path string is created as follows:

DLE base path: "C:\"

DLE current path: "DOCS\GRFS\DESIGN"

sname: "*.*"

GRFS agent creates the path string: "C:\DOCS\GRFS\DESIGN*.*"

The GRFS agent does not insert a path delimeter after the DLE base path because the DLE base path already ends with a path delimeter.

3.4.2 GRFS Find Info Area

One of the most important fields in the GRFS DBLK data area is the

Find Info area. Operating systems usually require some data which was returned from a FindFirst operation in order to perform subsequent FindNext operations. GRFS is designed so that the Find Info will reside in the GRFS DBLK, and the Find Info will be available to the GRFS agent whenever the GRFS_FIND_NEXT_OBJ command is issued. This is accomplished by passing the DBLK containing the Find Info back and forth between the backup application and the GRFS agent.

The backup application will never modify the Find Info data area.

The GRFS_FIND_NEXT_OBJ message has only two parameters:

dle_id: This field contains the id of the DLE that the backup application wishes to continue scanning.

dblk: This field is a DBLK which contains the Find Info data required for the agent to perform a FindNext operation.

The GRFS agent must respond with a GRFS_FIND_OBJ_STAT response message to both the GRFS_FIND_FIRST_OBJ and the GRFS_FIND_NEXT_OBJ commands. The parameters within this response message are described below:

more_flag: This field contains a boolean value that can be used by the GRFS agent to indicate to the GRFS file system whether there are any more objects available after the object currently being returned. If the more_flag is set to 0 (FALSE), then the next time the backup application makes a FindNextObject function call, the GRFS file system will immediately return FS_NO_MORE and will not a transmit GRFS_FIND_NEXT_OBJ command to the GRFS agent. If the agent is unable to know in advance if the object being returned is the last object available, then the agent can always set this field to a non-zero (TRUE) value. This will force the GRFS file system to send a GRFS_FIND_NEXT_OBJ command and the GRFS agent to respond with a FS_NO_MORE return value.

dblk: This field must be a complete GRFS DBLK. If a directory object is being returned, then the directory name should be a full path relative to the DLEs base path. For example, if the current path of a DLE is "OS2/SYSTEM", and the agent is returning the directory "TRACE", then the path returned in the DBLK data area would be "OS2\SYSTEM\TRACE". The path must be null-terminated, and the null-terminator character must be included in the path length field in the DBLK common structure. Root directory objects are returned with the path name '\0' and the path-leng field set to 1.

File object names are also returned as null-terminated strings, but only the actual file name is returned.

3.5 GRFS_FIND_CLOSE and GRFS_FIND_CLOSE_STAT

The GRFS_FIND_CLOSE command is used by the backup application when it is done scanning a particular directory. When a GRFS agent receives a GRFS_FIND_CLOSE message, the agent is allowed to release any resources associated with the FindFirst/FindNextfunctions. The are two parameters in the GRFS_FIND_CLOSE message and they are described below:

dle_id:

dblk:

The proper response message type for the GRFS_FIND_CLOSE command is the GRFS_FIND_CLOSE_STAT message. There are no parameters associated with the GRFS_FIND_CLOSE message.

3.6 GRFS_GET_OBJ_INFO, GRFS_GET_OBJ_INFO_STAT

The GRFS_GET_OBJ_INFO command is used by the backup application to retrieve a completed DBLK when the backup application has only a partially complete DBLK. The only DBLK fields which are required to contain valid data when the DBLK is passed to the GRFS agent are the blk_type (DIR or FILE) and the object name in the DBLK data area. The proper response message type is GRFS_GET_OBJ_INFO_STAT. The only parameter in the response message is the fully completed DBLK.

There is one slight difference between how a DBLK is created for the GRFS_GET_OBJ_INFO command. All other GRFS commands which create DBLKs return a fully specified path as the object name for directory objects. The GRFS_GET_OBJ_INFO_STAT DBLK returns ONLY the directory name as the path data in the DBLK data area. This is a "truth".

**** If the DBLK sent to the agent contains a Find Info area, then the agent MUST preserve this data within the DBLK which is returned to the backup application.

3.7 GRFS_GET_CURRENT_DDB, GRFS_GET_CURRENT_DDB_STAT

The GRFS_GET_CURRENT_DDB command is used by the backup application to retrieve a DBLK corresponding to the DLEs current directory path. The proper response message type is GRFS_GET_OBJ_INFO_STAT. The directory path string returned in the DBLK must be a fully specified relative to the DLE's base path. An example is presented below:

DLE's base path: "C:\OS2"

DLE's current path: "WINOS2\SYSTEM"

The path string returned in the DBLK data area would be "WINOS2\SYSTEM". An example of the DLE's current path being the logical root directory is presented below:

DLE's base path: "C:\OS2"

DLE's current path: ""

The path string data returned in the DBLK data area would be a '\0' and the b.d.os_path_leng field would be set to 1.

**** Whenever a GRFS agent returns a logical root directory object DBLK, the DBLK data area path string should be set to '\0' and the b.d.os_path_leng field should be 1.

3.8 GRFS_CREATE_OBJ, GRFS_CREATE_OBJ_STAT

The GRFS_CREATE_OBJ command is used by the backup application during restore operations in order to create a file system object. The parameters associated with this command are the following:

dle_id: This parameter contains the DLE handle of the DLE where the object should be created.

dblk: This parameter is a complete DBLK and contains the type and the name of the object to be created.

Directory object DBLKs will contain fully specified paths, so the DLE's current path is NOT included when creating the full path of the object to be created, GRFS Agents must be capable of creating all levels of a fully specified directory path from a single GRFS_CREATE_OBJ command. For example, the backup application may send the command to create the directory "WIN31\WORD\DOCS\ISPECS". If the any of the directories "DOCS", "WORD", or "WIN31" do not already exist, then the agent must first create the preceding directories within the fully specified path.

File objects are always created in the DLE's current path directory.

The proper response message type is GRFS_CREATE_OBJ_STAT. There are no parameters associated with this response message.

3.9 GRFS_OPEN_OBJ, GRFS_OPEN_OBJ_STAT

The backup application must "open" a file system object before any read, write or verify operations can be performed on the object. The three parameters associated with the GRFS_OPEN_OBJ command are described below:

dle_id: This field contains the DLE handle of the DLE where the object to be opened resides.

mode: This field contains a flag value which is GRFS agent must use to determine the mode which should be used to open the object. This value will be one of the following:

0	READ mode	(backup operation)
1	WRITE mode	(restore operation)
2	VERIFY mode	(compare operation)

dblk: This parameter is a complete DBLK and contains the type and the name of the object to be opened.

When a backup application is backing up a GRFS agent, the backup application may desire to backup files which are already in use on the GRFS agent's machine. The BEC_BACKUP_FILES_INUSE flag in the bec_flags field of the GRFS_ATTACH_DLE command determines whether the GRFS agent should attempt to open objects which have already been opened by a different process. If the DLE is configured to backup files in use and the agent is able to open the object, then the GRFS response message return code should be set to FS_OPENED_INUSE.

When an object is opened successfully, two parameters are returned in the GRFS_OPEN_OBJ_STAT response message. The first parameter is the obj_id. This parameter is a 32-bit value generated by the GRFS agent as an object handle. All succeeding GRFS commands which access the object will reference the obj_id. As with DLE handle ids, GRFS agents can use whatever method desired to generate the object handle ids.

A completed DBLK is also returned to the backup application in the response message. If the GRFS agent's operating system platform has any OS specific object attributes which are accessible only after the object has been successfully opened, they can be saved in the OS specific area within the DBLK's data area. One example of this is OS/2 "longnames" are accessible only after the object is opened.

3.10 GRFS_READ_OBJ, GRFS_READ_OBJ_STAT

The backup application uses the GRFS_READ_OBJ command to read data from previously opened file system objects. The parameters associated with this command are described below:

obj_id: This field contains the object handle id which was returned by the agent in the GRFS_OPEN_OBJ_STAT response message.

size: This field contains the size (in§p1230Xbytes) buffer which is available to receive data. The GRFS agent should endeavor to return as much data as possible for each GRFS_READ_OBJ command.

offset: This field contains the number of bytes offset into the object the agent should begin returning data from.

The proper response message type is GRFS_READ_OBJ_STAT. The response message has four fields which are described below:

size: This field should contain the actual number of bytes of data being returned in the response message.

blk_size: This field should usually be set to 1. This field is used by GRFS agents to request a specific number of bytes to be read by the next GRFS_READ_OBJ command. This functionality can be used if certain data areas must be read as "atomic" objects.

As an example, suppose the backup application requests to read 20 bytes. The GRFS agent has 14 bytes available, and then the next 12 bytes must be read a unit. The GRFS would return the 14 bytes, set the size field to 14, and set the blk_size field to 12. This will force the backup application to request 12 bytes in the next GRFS_READ_OBJ command.

The GRFS agent must never set the blk_size field larger than the negotiated GRFS maximum object buffer size.

strm_info: This field is a STREAM_INFO structure and is discussed in section 1.3 of this document.

data: This field is the buffer which contains the actual data. The size of this buffer is limited to the maximum object buffer size as negotiated during the DLB attach operation.

3.11 GRFS_WRITE_OBJ, GRFS_WRITE_OBJ_STAT

The backup application uses the GRFS_WRITE_OBJ command to restore data to a GRFS agent. The parameters associated with this command are described below:

obj_id: This field contains the object handle id which was returned by the agent in the GRFS_OPEN_OBJ_STAT response message.

size: This field contains the size (in bytes) of the data buffer which is to be written.

offset: This field contains the offset in bytes, from the beginning of the object, that the GRFS agent should begin writing the data buffer.

strm_info: This field contains a STREAM_INFO structure. As described for the GRFS_READ_OBJ response message, the first block of each data stream will have the strm_info.id field set to the stream data type. All succeeding blocks of that data stream type will have the strm_info.id field set to STRM_INVALID. The first block of a particular stream data type will have the strm_info.size field set to the total size (in bytes) of the stream.

GRFS agents should ignore a data block for a stream type that they do not recognize, and their response message should indicate that the entire block was successfully written.

data: This field is the buffer which contains the data block that is to be written.

The proper response message type is GRFS_WRITE_OBJ_STAT. This response message has the following parameters associated with it:

size: This field should be set to the number of bytes successfully written.

blk_size: This field should normally be set to 1. This field is used to indicate that the GRFS agent requires a specific number of bytes to be written in the next GRFS_WRITE_OBJ command. Any value other than 1 will force the backup application to attempt to write the requested number of bytes during the next GRFS_WRITE_OBJ operation. The agent should NEVER set this field to greater than the negotiated maximum object buffer size.

3.12 GRFS_VERIFY_OBJ, GRFS_VERIFY_OBJ_STAT

The backup application uses the GRFS_VERIFY_OBJ command to verify that data contained on the backup media matches the data residing on the GRFS agent. The parameters associated with this command are described below:

obj_id: This field contains the object handle id which was returned by the agent in the GRFS_OPEN_OBJ_STAT response message.

size: This field contains the size (in bytes) of the data buffer which is to be compared.

offset: This field contains the offset in bytes, from the beginning of the object, that the GRFS agent should begin comparing the data buffer.

strm_info: This field contains a STREAM_INFO structure and is described in section 1.3 of this document.

data: This field is the buffer which contains the data block that is to be verified.

The proper response message type is GRFS_VERIFY_OBJ_STAT. This response message has the following parameters associated with it:

size: This field should be set to the number of bytes successfully verified.

blk_size: This field should normally be set to 1. This field is used to indicate that the GRFS agent requires a specific number of bytes to be verified in the next GRFS_VERIFY_OBJ command. Any value other than 1 will force the backup application to attempt to verify the requested number of bytes during the next GRFS_VERIFY-OBJ operation. The agent should NEVER set this field to greater than the negotiated maximum object buffer size.

3.13 GRFS_SEEK_OBJ, GRFS_SEEK_OBJ_STAT

The backup application uses the GRFS_SEEK_OBJ command to force the GRFS agent to move the previously opened object's file location pointer to a specific offset within the object. This command is typically used by the backup application to seek past sectors which are unreadable in hopes that some of the data may be readable (HaHa). The parameters associated with this command are described below:

obj_id: This field contains the object handle id which was returned by the agent in the GRFS_OPEN_OBJ_STAT response message.

offset: This field contains the offset in bytes, from the beginning of the object, that the GRFS agent should move the file pointer to.

The proper response message type is GRFS_SEEK_OBJ_STAT. This response message contains only one parameter associated with it. The parameter, seek_obj_offset specifies the offset within the object that the agent was able to seek to.

3.14 GRFS_CLOSE_OBJ, GRFS_CLOSE_OBJ_STAT

The backup application uses the GRFS_CLOSE_OBJ command to force the GRFS agent to close a previously opened file system object. When an object is closed, the agent is allowed to free any resources associated with the open object. The only parameter in this command message is the obj_id field. This field contains the object handle id which was returned by this agent in the GRFS_OPEN_OBJ_STAT response message.

The proper response message type is GRFS_CLOSE_OBJ_STAT. There are no parameters with this response message.

3.15 GRFS_DELETE_OBJ, GRFS_DELETE_OBJ_STAT

The GRFS_DELETE_OBJ command is used by the backup application during transfer operations in order to remove a file system object. The parameters associated with this command are the following:

dle_id: This parameter contains the DLE handle of the DLE where the object should be removed.

dblk: This parameter is a complete DBLK, and contains the type and the name of the object to be deleted.

*p905Xfully Directory object DBLKs will contain specified paths, so the DLE's current path is NOT included when creating the full path of the object to be deleted. The backup application will first remove file objects from a directory object before removing the directory object.

File objects are always deleted from the DLE's current path directory.

The proper response message type is GRFS_DELETE_OBJ_STAT. There are no parameters associated with this response message.

3.16 GRFS_CHANGE_DIR, GRFS_CHANGE_DIR_STAT

The GRFS_CHANGE_DIR command is used by the backup application to force a GRFS agent to change the "current directory" of a specific DLE. The new path supplied in the message is always a fully specified path relative to the DLE's base path. The GRFS agent MUST verify that the new path is a valid path. This can usually be accomplished by performing a "FindFirst" operation on the new path. As an added bonus, the backup application may send a "null-impreguated" string in the path field. This means that the GRFS agent must replace the internal '\0' path delimeters with the agent's OS specific path delimeter character. No applause necessary.

The proper response message type is GRFS_CHANGE_DIR_STAT. There are no parameters associated with this response message.

3.18 GRFS_SET_OBJ_INFO, GRFS_SET_OBJ_INFO_STAT

The GRFS_SET_OBJ_INFO command is used by the backup application to set the file system attributes of a file system object. The parameters associated with this command are described below:

dle_id: This parameter contains the DLE handle id of the DLE where the object resides.

dblk: This parameter is complete DBLK and contains the object type, the object name, and the object attribute data which are to be set.

The GRFS agent must set the following file system object attributes:

ctime	(CREATION TIME)	
atime	(ACCESS TIME)	(if possible)
time	(MODIFIED TIME)	
size	(object data size)	
gen_attr	(file system attribute flags)	

The proper response message type is GRFS_SET_OBJ_INFO_STAT. There are no parameters associated with this response message.

3.19 GRFS_VERIFY_OBJ_INFO, GRFS_VERIFY_OBJ_INFO_STAT

The GRFS_VERIFY_OBJ_INFO command is used by the backup application to verify that file system object attributes on the GRFS agent match the object attributes contained on the backup media. The parameters associated with this command are described below:

dle_id: This parameter contains the DLE handle of the DLE where the object resides.

dblk: This parameter is a complete DBLK and contains the object type, the object name, and the object attribute data which are to be compared.

The GRFS agent must verify that the following input parameter DBLK fields match the actual attributes of the file system object:

cdate	(CREATION DATE)
mdate	(MODIFIED DATE)
size	(object data size)
gen_attr	(file system attribute flags)

The proper response message type is GRFS_VERIFY_OBJ_INFO_STAT. There are no parameters associated with this response message.

3.20 GRFS_PREPARE_DBLK, GRFS_PREPARE_DBLK_STAT

The GRFS_PREPARE_DBLK command is used so that during restore operations the GRFS Agent is able to modify ("image") path and directory names into a form which is usable by the target (restore) agent's file systems. For instance, if a backup set is created by a MacIntosh agent, then the file and directory names must be modified in order to restore the backup set onto a DOS agent's FAT file system 8.3 format.

dle_id: This parameter contains the DLE handle of the DLE where the object resides.

dblk: This parameter is a complete DBLK and contains the object type, the object name.

The agent should append the modified name at the end of the DBLK and alter the "os_" name pointers to point to the new name. The agent must also modify the dblk.dblk_actual_size to account for the increased DBLK size. If the input name does not require modification, then the DBLK can be returned unmodified.

Appendix A - GRFS Technical Reference

This section of the GRFS Technical Reference appendix shows the actual definitions of the structures which have been described in this document. All of the structures can be found the GRFS.H include file.

```
typedef union
        {
        INT8            val[4];
        INT32           num;
        } INET32;

typedef union
        {
        UINT8           val[4];
        UINT32          num;
        } UNET32;

typedef union
        {
        INT8            val[2];
        INT16           num;
        } INET16;

typedef union
        {
        UINT8           val[2];
        UINT16          num;
        } UNET16;

typedef struct
        {
        UNET32          lsw;
        UNET32          msw;
        } UNET64;

typedef UNET 32 DLE_HANDLE;
typedef UNET32 OBJ_HANDLE;
typedef UNET32 REQ_HANDLE;

GENERIC DBLK NETWORK STRUCTURE

struct grfs_gen_dblk_str
        {
        UINT8           blk_type;
        UINT8           res1;
        UINT8           fg_com_reserve[38];

struct STD_OBJ_INFO
```

```
        {
        UINT8         os_id;
        UINT8         os_ver;
        UINT8         res2[2];
        DATE_TIME     ctime;
        DATE_TIME     atime;
        DATE_TIME     btime;
        DATE_TIME     time;
        UNET64        size;
        UNET32        gen_attr;
        } std_info;

BOOLEAN       os_info_complete;
UNET16        min_ddb_info;
UNET16        min_ddb_size;
UNET16        os_spec_info;
UNET16        os_spec_size;
UNET16        dblk_actual_size;
UNET16        tape_attribs;
UNET16              name_complete;
UNET16        find_info;
UNET16        find_info_size;
BOOLEAN       translate_flag;
BOOLEAN       special_flag;
UINT8         obj_type;
union
        {
        struct OS_DDB_INFO
              {
              UNET16        os_path;
              UNET16        os_path_leng;
              UNET16        path_leng;
              UNET16        path;
              } d;
        struct OS_FDB_INFO
              {
              BOOLEAN       inuse_attrib;
              UNET16        os_name;
              UNET16        name;
              } f;
        } b;
};

typedef   struct   grfs_gen_dblk_str   GRFS_GEN_DBLK,
*GRFS_GEN_DBLK_PTR;

struct grfs_message
        {
        UINT8         msg_type;
        UINT8         reserved;
        UINT16        retcode;
```

```
UNET32        request_id
union {
      /** GRFS command parameter structures **/
      DLE_HANDLE                        dle_id;
      OBJ_HANDLE                        obj_id;
      GRFS_ATTACH_DLE_PARMS             attach_parms;
      GRFS_FIND_FIRST_OBJ_PARMS         ff_obj_parms;
      GRFS_OBJECT_PARMS                 obj_parms;
      GRFS_OPEN_OBJ_PARMS               open_obj_parms;
      GRFS_READ_OBJ_PARMS               read_obj_parms;
      GRFS_WRITE_OBJ_PARMS              write_obj_parms;
      GRFS_VERIFY_OBJ_PARMS             verify_obj_parms;
      GRFS_SEEK_OBJ_PARMS               seek_obj_parms;
      GRFS_CHANGE_DIR_PARMS             change_dir_parms;
      GRFS_ENUM_SPEC_PARMS              enum_spec_parms;

      /** GRFS response parameter structures **/
      UNET32                            seek_obj_offset;
      GRFS_GEN_DBLK                     dblk;
      GRFS_ATTACH_DLE_STAT_PARMS        attach_stat;
      GRFS_FIND_DLE_STAT_PARMS          find_dle_stat;
      GRFS_FIND_OBJ_STAT_PARMS          find_obj_stat;
      GRFS_OPEN_OBJ_STAT_PARMS          open_obj_stat;
      GRFS_READ_OBJ_STAT_PARMS          read_obj_stat;
      GRFS_WRITE_OBJ_STAT_PARMS         write_obj_stat:
      GRFS_VERIFY_OBJ_STAT_PARMS        verify_obj_stat;
      GRFS_ENUM_SPEC_STAT_PARMS         enum_special_stat;
      } msg_parms;
};
```

This section shows the GRFS command message types and their corresponding GRFS response message types. The parameters associated with each message are also provided.

GRFS COMMAND MESSAGES		GRFS RESPONSE MESSAGES	
GRFS_ATTACH_DLE(	dle_name[], bee_flags, special_word, max_obj_bsize, dle_parent, cmpr_type, user_name[], password[])	GRFS_ATTACH_DLE_STAT(	dle_id, max_connects, max_opens_per_connect, process_ddbs, max_obj_bsize, cmpr_type, supports_children path_len, current_path[])
GRFS_FIND_FIRST_DLE(	dle_id)	GRFS_FIND_DLE_STAT(	dle_name[], path_delim, passwd_req, user_req, dle_writeable, supports_last_access, os_id, os_ver, {s_type, crypt_type, cmpr_type, more_flag)
GRFS_FIND_NEXT_DLE(	dle_id)	GRFS_FIND_DLE_STAT(	dle_name[], path_delim, passwd_req, user_req, dle_writeable, os_id, os_ver, fs_type, crypt_type, cmpr_type, more_flag)
GRFS_DETACH_DLE(	dle_id)	GRFS_DETACH_DLE_STAT(	---)
GRFS_FIND_FIRST_OBJ(	dle_id, find_type, sname[])	GRFS_FIND_OBJ_STAT(	more_flag, dblk)
GRFS_FIND_NEXT_OBJ(	dle,id, dblk)	GRFS_FIND_OBJ_STAT(	more_flag, dblk)
GRFS_FIND_CLOSE(	dle_id, dblk)	GRFS_FIND_CLOSE_STAT(	---)
GRFS_CREATE_OBJ(	dle_id, dblk)	GRFS_CREATE_OBJ_STAT(	---)
GRFS_OPEN_OBJ(	dle_id, mode,	GRFS_OPEN_OBJ_STAT(	obj_id, dblk)

```
                    dblk)

GRFS_READ_OBJ(      obj_id,        GRFS_READ_OBJ_STAT(      size,
                    size,                                   blk_size,
                    offset)                                 strm_info,
                                                            buffer[])

GRFS_WRITE_OBJ(     obj_id,        GRFS_WRITE_OBJ_STAT(     size,
                    size,                                   blk_size)
                    offset,
                    strm_info,
                    buffer[])

GRFS_SEEK_OBJ(      obj_id,        GRFS_SEEK_OBJ_STAT( seek_obj_offset)
                    offset)

GRFS_VERIFY_OBJ(    obj_id,        GRFS_VERIFY_OBJ_STAT(    size,
                    size,                                   blk_size)
                    offset,
                    strm_info,
                    buffer[])

GRFS_CLOSE_OBJ(     obj_id)        GRFS_CLOSE_OBJ_STAT(     ---)

GRFS_DELETE_OBJ(    dle_id,        GRFS_DELETE_OBJ_STAT(    ---)
                    dblk)

GRFS_GET_OBJ_INFO(dle_id,          GRFS_GET_OBJ_INFO_STAT( dblk)
                    dblk)

GRFS_VERIFY_OBJ_INFO( dle_id,      GRFS_VERIFY_OBJ_INFO_STAT( ---)
                    dblk)

GRFS_CHANGE_DIR(    dle_id,        GRFS_CHANGE_DIR_STAT(    ---)
                    net_path[],
                    size)

GRFS_GET_CUR_DDB( dle_id_)         GRFS_GET_CUR_DDB_STAT(   dblk)

GRFS_SET_OBJ_INFO(dle_id,          GRFS_SET_OBJ_INFO_STAT( ---)
                    dblk)

GRFS_ENUM_SPECIAL_FIRST            GRFS_ENUM_SPECIAL_STAT( name[],
                    ( dle_id,                               more_flag)
                    enum_type)

GRFS_ENUM_SPECIAL_NEXT             GRFS_ENUM_SPECIAL_STAT( name[],
                    (dle_id,                                more_flag)
                    enum_type)

GRFS_SPECIAL_EXCLUDE               GRFS_SPECIAL_EXCLUDE_STAT(---)
                    (path_len,
                    fname_len,
                    data[])

GRFS_PREPARE_DBLK                  GRFS_PREPARE_DBLK_STAT( dblk)
                    (dle_id,
                    dblk)
```

COMMON GRFS MESSAGE PROCESSING

All GRFS messages generated by the backup application include the following common fields: *msg_type*, *retcode*, and *request_id*. The msg_type field must contain a valid GRFS command value. The backup application will set the *request_id* field to a value which the backup application will use to correlate outgoing GRFS command messages to the corresponding incoming GRFS response messages. The GRFS agent must set the request_id value of the GRFS response message to the request_id value received in the corresponding GRFS command message. The GRFS response message to the request_id value received in the corresponding GRFS command message. The *ret_code* field is not used for GRFS command messages; it is meaningful only for GRFS response messages.

Several of the message parameter structures contain large fields (DBLKs, full-path names) which are defined statically but contain variable length data, and these data fields will typically fill only a small portion of the allotted space. These large fields are always declared as the last member in the parameter structure. Only the portion of the message parameter field which is actually used must be transmitted across the network. This will allow the GRFS to be more efficient because most non object-data GRFS messages can be transmitted as a single NRL transport packet.

CRITICAL ERROR HANDLING

GRFS agent programs must handle critical error situations without hanging the agent's system. When a GRFS agent detects a critical error while performing an GRFS command, the agent program should "fail" the operation and set the <u>retcode</u> field appropriately (FS_DEVICE_ERROR, etc). The agent can also retry the failed operation before returning a GRFS status message to the backup application. When a fatal FS error code is returned to the backup application, the application user will be given the opportunity to decide whether to retry the failed operation.

GRFS Messages Type Values

GRFS COMMANDS	
GRFS_ATTACH_DLE	0x01
GRFS_FIND_FIRST_DLE	0x02
GRFS_FIND_NEXT_DLE	0x03
GRFS_DETACH_DLE	0x04
GRFS_FIND_FIRST_OBJ	0x05
GRFS_FIND_NEXT_OBJ	0x06
GRFS_FIND_CLOSE	0x07
GRFS_CREATE_OBJ	0x08
GRFS_OPEN_OBJ	0x09
GRFS_READ_OBJ	0x0A
GRFS_WRITE_OBJ	0x0B
GRFS_SEEK_OBJ	0x0C
GRFS_VERIFY_OBJ	0x0D
GRFS_CLOSE_OBJ	0x0E
GRFS_DELETE_OBJ	0x0F
GRFS_GET_OBJ_INFO	0x10
GRFS_VERIFY_OBJ_INFO	0x11
GRFS_CHANGE_DIR	0x12
GRFS_GET_CUR_DDB	0x13
GRFS_SET_OBJ_INFO	0x14
GRFS_ENUM_SPECIAL_FIRST	0x15
GRFS_ENUM_SPECIAL_NEXT	0x16
GRFS_SPECIAL_EXCLUDE	0x17
GRFS_PREPARE_DBLK	0x18

GRFS RESPONSES	
GRFS_ATTACH_DLE_STAT	0x41
GRFS_FIND_DLE_STAT	0x42
GRFS_DETACH_DLE_STAT	0x44
GRFS_FIND_OBJ_STAT	0x45
GRFS_FIND_CLOSE_STAT	0x47
GRFS_CREATE_OBJ_STAT	0x48
GRFS_OPEN_OBJ_STAT	0x49
GRFS_READ_OBJ_STAT	0x4A
GRFS_WRITE_OBJ_STAT	0x4B
GRFS_SEEK_OBJ_STAT	0x4C
GRFS_VERIFY_OBJ_STAT	0x4D
GRFS_CLOSE_OBJ_STAT	0x4E
GRFS_DELETE_OBJ_STAT	0x4F
GRFS_GET_OBJ_INFO_STAT	0x50
GRFS_VERIFY_OBJ_INFO_STAT	0x51
GRFS_CHANGE_DIR_STAT	0x52
GRFS_GET_CUR_DDB_STAT	0x53
GRFS_SET_OBJ_INFO_STAT	0x54
GRFS_ENUM_SPECIAL_STAT	0x55
GRFS_SPECIAL_EXCLUDE_STAT	0x57
GRFS_PREPARE_DBLK_STAT	0x58

GRFS COMMAND MESSAGES	MESSAGE PARAMETER STRUCTURE

GRFS_ATTACH_DLE	struct GRFS_ATTACH_DLE_PARMS
	{
	CHAR dle_name[GRFS_MAX_DLE_NAME_LEN];
	INET16 bec_flags
	INET16 special_word;
	UNET16 max_obj_bsize;
	DLE_HANDLE dle_parent;
	UINTE8 cmpr_type;
	CHAR user_name[48];
	CHAR password[MAX_PASSOWRD_LEN];
	};

GRFS_FIND_FIRST_DLE	DLE_HAND dle_id;
GRFS_FIND_NEXT_DLE	DLE_HAND dle_id;
GRFS_DETACH_DLE	DLE_HAND dle_id;

GRFS_FIND_FIRST_OBJ	struct GRFS_FIND_FIRST_OBJ_PARMS
	{
	DLE_HAND dle.id;
	UNET16 find_type;
	CHAR sname[GRFS_MAX_SNAME];
	};

GRFS_FIND_NEXT_OBJ
GRFS_FIND_CLOSE
GRFS_CREATE_OBJ
GRFS_DELETE_OBJ
GRFS_GET_OBJ_INFO
GRFS_VERIFY_OBJ_INFO
GRFS_SET_OBJ_INFO
	struct GRFS_OBJECT_PARMS
	{
	DLE_HAND dle_id;
	GRFS_GEN_DBLK dblk;
	};

GRFS_OPEN_OBJ	struct GRFS_OPEN_OBJ_PARMS
	{
	DLE_HAND dle_id;
	INET16 mode;
	UNIT8 reserved[2];
	GRFS_GEN_DBLK dblk;
	};

GRFS_READ_OBJ	struct GRFS_READ_OBJ_PARMS
	{
	OBJ_HAND obj_id;
	UNET16 size;
	UNET32 offset;
	};

GRFS_WRITE_OBJ	struct GRFS_WRITE_OBJ_PARMS {

```
                               OBJ_HAND             obj_id;
                               UNET32               offset;
                               STREAM_INFO          strm_info;
                               UNET16               size;
                         UINT8 buffer[GRFS_MIN_OBJ_SIZE];
                               };

GRFS_SEEK_OBJ            struct GRFS_SEEK_OBJ_PARMS
                               {
                               OBJ_HAND             obj_id;
                               UNET32               offset
                               };

GRFS_VERIFY_OBJ          struct GRFS_VERIFY_OBJ_PARMS
                               {
                               OBJ_HAND             obj_id;
                               UNET32               offset;
                               STREAM_INFO          strm_info;
                               UNET16               size;
                         UINT8 buffer[GRFS_MIN_OBJ_SIZE];
                               };

GRFS_CLOSE_OBJ           OBJ_HAND    obj_id

GRFS_CHANGE_DIR          struct GRFS_CHANGE_DIR_PARMS
                               {
                               DLE_HAND     dle_id;
                               INET16       size;
                         CHAR net_path{GRFS_MAX_PATH_LEN];
                               };

GRFS_ENUM_SPECIAL_FIRST  struct GRFS_ENUM_SPEC_PARMS
GRFS_ENUM_SPECIAL_NEXT         {
                               DLE_HAND             dle_id;
                               UNET16               enum_type;
                               };

GRFS_SPECIAL_EXCLUDE     struct GRFS_SPEC_EXCLUDE_PARMS
                               {
                               INET16               path_len;
                               INET16               fname_len;
                         UINT8 buffer[GRFS_MIN_OBJ_SIZE];
                               };
```

GRFS RESPONSE MESSAGES	MESSAGE PARAMETER STRUCTURE

GRFS_ATTACH_DLE_STAT	struct GRFS_ATTACH_DLE_STAT_PARMS
	{
	DLE_HAND dle_id;
	INET16 max_connects;
	INET16 max_opens_per_connect;
	UNET16 process_ddbs;
	INET16 max_obj_bsize;
	BOOLEAN supports_children;
	UNET16 path_len
	UINT8 cmpr_type;
	CHAR current_path[GRFS_MAX_PATH_LEN];
	};

GRFS_FIND_DLE_STAT	struct GRFS_FIND_DLE_STAT_PARMS
	{
	CHAR dle_name[GRFS_MAX_DLE_NAME_LEN];
	CHAR path_delim;
	UINT8 res1;
	BOOLEAN passwd_req;
	BOOLEAN user_req;
	BOOLEAN dle_writeable;
	BOOLEAN last_access_supported;
	INT8 os_id;
	INT8 os_ver;
	INET16 fs_type;
	UINT8 crypt_type;
	UINT8 cmpr_type;
	BOOLEAN more_flag
	};

GRFS_DETACH_DLE_STAT	none

GRFS_FIND_OBJ_STAT	struct GRFS_FIND_OBJ_STAT_PARMS
	{
	BOOLEAN more_flag;
	UINT8 reserved[2];
	GRFS_GEN_DBLK dblk;
	};

GRFS_FIND_CLOSE_STAT	none
GRFS_CREATE_OBJ_STAT	none

GRFS_OPEN_OBJ_STAT	struct GRFS_OPEN_OBJ_STAT_PARMS
	{
	OBJ HAND obj.id;
	GRFS_GEN_DBLK dblk;
	};

GRFS_READ_OBJ_STAT	struct GRFS_READ_OBJ_STAT_PARMS {

	`UNET16 size;`
	`UNET16 blk_size;`
	`STREAM_INFO strm_info;`
	`UINT8 buffer[GRFS_MIN_OBJ_SIZE];`
	`};`

GRFS_WRITE_OBJ_STAT	`struct GRFS_WRITE_OBJ_STAT_PARMS`
	`{`
	`UNET16 size;`
	`UNET16 blk_size;`
	`};`

GRFS_SEEK_OBJ_STAT	`UNET32 offset`

GRFS_VERIFY_OBJ_STAT	`struct GRFS_VERIFY_OBJ_STAT_PARMS`
	`{`
	`UNET16 size;`
	`UNET16 blk_size;`
	`};`

GRFS_CLOSE_OBJ_STAT	none
GRFS_DELETE_OBJ_STAT	none
GRFS_GET_OBJ_INFO_STAT	`GRFS_GEN_DBLK dblk;`
GRFS_VERIFY_OBJ_INFO_STAT	none
GRFS_CHANGE_DIR_STAT	none
GRFS_GET_CUR_DDB_STAT	`GRFS_GEN_DBLK dblk;`
GRFS_SET_OBJ_INFO_STAT	none

GRFS_ENUM_SPECIAL_STAT	`struct GRFS_ENUM_SPECIAL_STAT_PARMS`
	`{`
	`BOOLEAN more.flag;`
	`INET16 path_len;`
	`INET16 fname_len;`
	`UINT8 buffer[GRFS_MIN_OBJ_SIZE];`
	`};`

GRFS RETURN CODES

The following values have been defined for GRFS agents to use as return codes in the retcode field of GRFS response messages:

SUCCESS	0x0000
OUT_OF_MEMORY	0xFFFF
FS_NEVER_ATTACHED	0xFE01
FS_BAD_DBLK	0xFE02
FS_DLE_NOT_ATTACHED	0xFE03
FS_STACK_EMPTY	0xFE04
FS_ACCESS_DENIED	0xFE05
FS_OUT_OF_SPACE	0xFE06
FS_NO_MORE	0xFE07
FS_NOT_FOUND	0xFE08
FS_INVALID_DIR	0xFE09
FS_AT_ROOT	0xFE0A
FS_OBJECT_NOT_OPENED	0xFE0B
FS_EOF_REACHED	0xFE0C
FS_DEVICE_ERROR	0xFE0D
FS_GDATA_DIFFERENT	0xFE0E
FS_SECURITY_DIFFERENT	0xFE0F
FS_OPENED_INUSE	0xFE10
FS_IN_USE_ERROR	0xFE11
FS_INFO_DIFFERENT	0xFE12
FS_BUFFER_TO_SMALL	0xFE13
FS_DEFAULT_SPECIFIED	0xFE14
FS_RESDATA_DIFFERENT	0xFE15
FS_INCOMPATIBLE_OBJECT	0xFE16
FS_NOT_INITIALIZED	0xFE17
FS_UNDEFINED_TYPE	0xFE18
FS_NOT_OPEN	0xFE19
FS_INVALID_DLE	0xFE1A
FS_NO_MORE_DLE	0xFE1B
FS_BAD_DLE_HAND	0xFE1C
FS_DRIVE_LIST_ERROR	0xFE1D
FS_ATTACH_TO_PARENT	0xFE1E
FS_DEVICE_NOT_FOUND	0xFE1F
FS_BAD_INPUT_DATA	0xFE20
FS_OS_ATTRIB_DIFFER	0xFE21
INVALID_PATH_DESCRIPTOR	0xFE22
INVALID_FILE_DESCRIPTOR	0xFE23
DRIVE_DESCRIPTOR_ERROR	0xFE24
FS_NO_MORE_CONNECTIONS	0xFE25
FS_SERVER_ADDR_NOT_FOUND	0xFE26
FS_MAX_SERVER_CONNECTIONS	0xFE27
FS_BAD_ATTACH_TO_SERVER	0xFE28
FS_BAD_SERVER_LOGIN	0xFE29
FS_SERVER_LOGOUT_DENIED	0xFE2A
FS_BAD_ATTR_READ	0xFE2B
FS_EADATA_DIFFERENT	0xFE2C
FS_OBJECT_CORRUPT	0xFE2D
FS_ACLDATA_DIFFERENT	0xFE2E
FS_CHILDREN_NOT_COMPLETE	0xFE2F
FS_COMM_FAILURE	0xFE30
FS_NET_DEV_ERROR	0xFE31
FS_DONT_WANT_STREAM	0xFEB1

The following section provides a list of likely return code values for each of the GRFS response messages. GRFS agents should use the return value listed above which provides the best indication for the cause of an error.

GRFS_ATTACH_DLE_STAT	
FS_ACCESS_DENIED	The user or password field was not valid.
FS_INVALID_DLE	The dle_name was invalid
OUT_OF_MEMORY	
GRFS_FIND_DLE_STAT	
FS_INVALID_DLE	dle_id was invalid
FS_NO_MORE	No more DLEs to enumerate
GRFS_DETACH_DLE_STAT	
FS_INVALID_DLE	dle_id was invalid
GRFS_FIND_OBJ_STAT	
FS_INVALID_DLE	dle_id was invalid
FS_NO_MORE	No more file system objects to enumerate
GRFS_FIND_CLOSE_STAT	
FS_INVALID_DLE	dle_id was invalid
GRFS_CREATE_OBJ_STAT	
FS_INVALID_DLE	dle_id was invalid
FS_DEVICE_ERROR	"hard" device error, unable to create object
FS_ACCESS_DENIED	Agent does not have permission to create object
FS_BAD_DBLK	The DBLK data is invalid
GRFS_OPEN_OBJ_STAT	
FS_OPENED_INUSE	Object already opened by another process, but not locked, and BEC_CONFIG flag BEC_BACKUP_FILES_IN_USE is set
FS_IN_USE_ERROR	Object already opened by another process and locked, BEC_BACKUP_FILES_IN_USE not set
FS_INVALID_DLE	dle_id was invalid
FS_NOT_FOUND	Object not found
FS_DEVICE_ERROR	"hard" device error, unable to open object
FS_BAD_DBLK	The DBLK data was invalid
FS_ACCESS_DENIED	Agent does not have permission to open object
OUT_OF_MEMORY	
GRFS_READ_OBJ_STAT	
FS_DEVICE_ERROR	"hard" device error read
FS_OBJECT_NOT_OPENED	obj_id parameter was invalid
FS_EOF_REACHED	End of File already reached
FS_ACCESS_DENIED	Agent does not have permission to read object
GRFS_WRITE_OBJ_STAT	
FS_OBJECT_NOT_OPENED	obj_id parameter not invalid
FS_DEVICE_ERROR	"hard" device write error

FS_OBJECT_NOT_OPENED obj_id parameter was invalid

FS_OUT_OF_SPACE	Device is full
FS_ACCESS_DENIED	Agent does not have permission to write object
FS_DONT_WANT_STREAM	Agent does not want to restore this data stream
GRFS_SEEK_OBJ_STAT	
FS_OBJECT_NOT_OPENED	obj_id parameter was invalid
FS_EOF_REACHED	End of File already reached
FS_DEVICE_ERROR	"hard" device seek error
GRFS_VERIFY_OBJ_STAT	
FS_OBJECT_NOT_OPENED	obj_id parameter was invalid
FS_DEVICE_ERROR	"hard" error
FS_EOF_REACHED	End of File already reached
FS_GDATA_DIFFERENT	Object's normal data stream does not match
FS_SECURITY_DIFFERENT	Object's security data stream does not match
FS_EADATA_DIFFERENT	Object's extended attribute data stream does not match
FS_DONT_WANT_STREAM	Agent does not support this data stream type
GRFS_CLOSE_OBJ_STAT	
FS_OBJECT_NOT_OPENED	obj_id parameter was invalid
FS_DEVICE_ERROR	"hard" error
GRFS_DELETE_OBJ_STAT	
FS_INVALID_DLE	dle_id was invalid
FS_NOT_FOUND	Object not found
FS_DEVICE_ERROR	"hard" device error, unable to delete object
FS_BAD_DBLK	The DBLK data was invalid
FS_ACCESS_DENIED	Agent does not have permission to delete object
GRFS_GET_OBJ_INFO_STAT	
FS_INVALID_DLE	dle_id was invalid
FS_NO_MORE	Object not found
FS_DEVICE_ERROR	"hard" device error, unable to delete object
FS_BAD_DBLK	The DBLK data was invalid
GRFS_VERIFY_OBJ_INFO_STAT	
FS_INVALID_DLE	dle_id was invalid
FS_NOT_FOUND	Object not found
FS_DEVICE_ERROR	"hard" device error, unable to scan device
FS_BAD_DBLK	The DBLK data was invalid
FS_INFO_DIFFERENT	The object's attributes do not match
GRFS_CHANGE_DIR_STAT	
FS_INVALID_DLE	dle_id was invalid
FS_INVALID_DIR	net_path □ too long, or new path does not exist
FS_DEVICE_ERROR	"hard" device error, unable to scan device

```
GRFS_GET_CUR_DDB_STAT
      FS_INVALID_DLE         dle_id was invalid
      FS_DEVICE_ERROR        "hard" device error, unable to scan
                             device

GRFS_SET_OBJ_INFO_STAT
      FS_INVALID_DLE         dle_id was invalid
```

DBLK Fields

The individual fields within the GRFS common DBLK structure which must be manipulated by GRFS agent programs are described below.

blk_type: Defines whether the object is a file or a directory.

files	= 08
directories	= 09

os_id;

os_ver;

ctime:
atime:
btime:
time: These four fields are all defined as type DATE_TIME structures. The DATE_TIME structure has the following format:

```
struct DATE_TIME {
  UINT16date_valid;      /*TRUE or FALSE  */
  UINT16year;            /*year since 1980  */
  UINT16month;           /* 1 to 12  */
  UINT16day;             /* 1 to 31  */
  UINT16hour;            /* 0 to 23  */
  UINT16minute;          /* 0 to 59  */
  UINT16second;          /* 0 to 59  */
  UINT16day_of_week;     /* 1 to 7 Sun to Sat  */
     };
```

ctime = Object CREATION time
atime = Object ACCESSED time
btime = Object ARCHIVED time
time = Object MODIFIED time

If the OS of GRFS Agent being developed does not support one or more of the specific time stamps, then those time stamp fields should be reset to all zeros.

size: The size field contains the size of the normal data associated with the object. For instance the OS/2 Agent does NOT include the size of EAs and ACLs associated with an object.

gen_attr: This field is a bit-mapped flag which describes the file system attributes of the object. The following flag values can be contained in this field:

FILE_NORMAL	0x0000
FILE_READONLY	0x0001
FILE_HIDDEN	0x0002
FILE_SYSTEM	0x0004
FILE_DIRECTORY	0x0010
FILE_ARCHIVED	0x0020

os_info_complete This field is a boolean value which must be set to TRUE when the all the DBLK information for an object has been filled in.

min_ddb_info	This field contains a pointer to the information in the DBLK data area which is required to perform either a GRFS_GET_OBJ_INFO or GRFS_FIND_NEXT_OBJ command. The information pointed to by this field must be contiguous within the data area. Typically the DBLK find information and the object name constitute the "MIN_DDB_INFO". The DBLK find information is described in the find_info DBLK field.
min_ddb_size	This field contains the number of bytes of data pointed to by the min_ddb_info field.
os_spec_info	This field contains a pointer to the DBLK data area which contains any OS specific information

that the GRFS agent would like preserved during backup and restoration operations. For instance the OS/2 agent uses this area to save HPFS "Long Names" when they are present. As another example, a Unix GRFS agent could use this field to save information about special device placeholder files.

os_spec_size	This field contains the number of bytes of data pointed to by the os_spec_info field.
dblk_actual_size	This field contains the size of the entire DBLK. This value is the sum of the size of the GRFS DBLK common structure and the number of bytes of data within the variable length DBLK data area. Remember that the total DBLK must at most 1024 bytes long.
tape_attribs	not used
find_info	This field contains a pointer to the information in DBLK data area which can be used by the GRFS agent to perform a GRFS_FIND_NEXT_OBJ command. Examples of this field are the DOS GRFS agent passing a DTA structure and the OS/2 agent passing the DosFindFirst0HDIR value.
find_info_size	This field contains the number of bytes of data pointed to by the find_info field.
obj_type	not used
translate_flag	not used
special_flag	not used
b.d.os_path	This field contains a pointer to the path string contained within the DBLK data area for a directory object. The path string should not begin with a path delimeter character unless it is the root directory of a DLE. The path string must be null-terminated. During backup operations the os_path field and the path field will be identical. During restore operations, the os_path field will represent the "source" path and the path field will represent the "destination" path.
b.d.os_path_leng	This field contains the length of the path pointed to by the os_path field. This value should include the null-termination character.
b.d.path_leng	This field contains the length of the path pointed to by the path field. This value should include the null-termination character.

b.d.path	This field contains a pointer to the path string contained within the DBLK data area for a directory object. The path string should not begin with a path delimeter character unless it is the root directory of a DLE. The path string must be null-terminated. During backup operations, the path field will be the same as the os_path field; however during restore operations the path field may be different than the os_path field.
b.d.inuse_attrib	This field contains a flag which is used to mark files which have been opened but the file is currently also opened by another process.
b.f.os_name	This field contains a pointer to the file name string contained within the DBLK data area for a file object. The path string must be null-terminated. The os_name field and the name field will be the same during backup operations. During restore operations the os_name field represents the "source" file name whereas the name field represents the "destination" file name.
b.f.name	This field contains a pointer to the file name string contained within the DBLK data area for a file object. The path string must be null-terminated.

**** Whenever a GRFS agent returns a DLE's logical root directory object DBLK, the DBLK data area path string should be set to '\0' and the b.d.os_path_leng field should be 1.

CLAIMS

What is claimed is:

1. A computer network, comprising:

a) a plurality of computers running disparate operating systems, respectively;

b) a storage device for backing up and restoring data processed on the network; and

c) means for performing backup to and restore from the storage device, including:

i) a GRFS file system running on one of the said computers;

ii) a plurality of GRFS agents each running on a respective one of said computers; and

iii) wherein said GRFS file system and each of said GRFS agents interface with one another via command and response messages, respectively, said command and response messages being structured to support the disparate operating systems.

2. A computer network, according to claim 1, wherein said disparate operating systems have different data structure alignments, and said command and response messages are structured with a least common denominator alignment for said disparate operating systems.

3. A computer network, according to claim 1, wherein said command and response messages are further structured to interchange data between said disparate operating systems.

4. A computer network, according to claim 3, wherein said interchange structure of said command and response messages enable data from one of said computers running one of said operating systems to backed up to said storage device and said backed up data to be restored to another of said computers running another of said disparate operating systems.

5. A computer network, according to claim 3, wherein said interchange structure of said messages includes a streamer header having an identification value determining whether an associated data stream type is supported by a given one of said disparate operating systems.

6. A computer network, according to claim 1, wherein said command and response messages are further structured to enable independent multiple users of said plurality of computers to request simultaneously backup or restore of the data.

7. A computer network, according to claim 6, wherein said command and response messages are structured with a request id and wherein said GRFS file system may create a unique request id for every GRFS command message, whereby the GRFS file system can communicate simultaneously with multiple GRFS agents.

8. A computer network, according to claim 1, wherein said plurality of computers each has a user interface to enable a user to select backup or restore of selected data.

9. A computer network, according to claim 1, wherein said network may have an additional computer not running a GRFS agent.

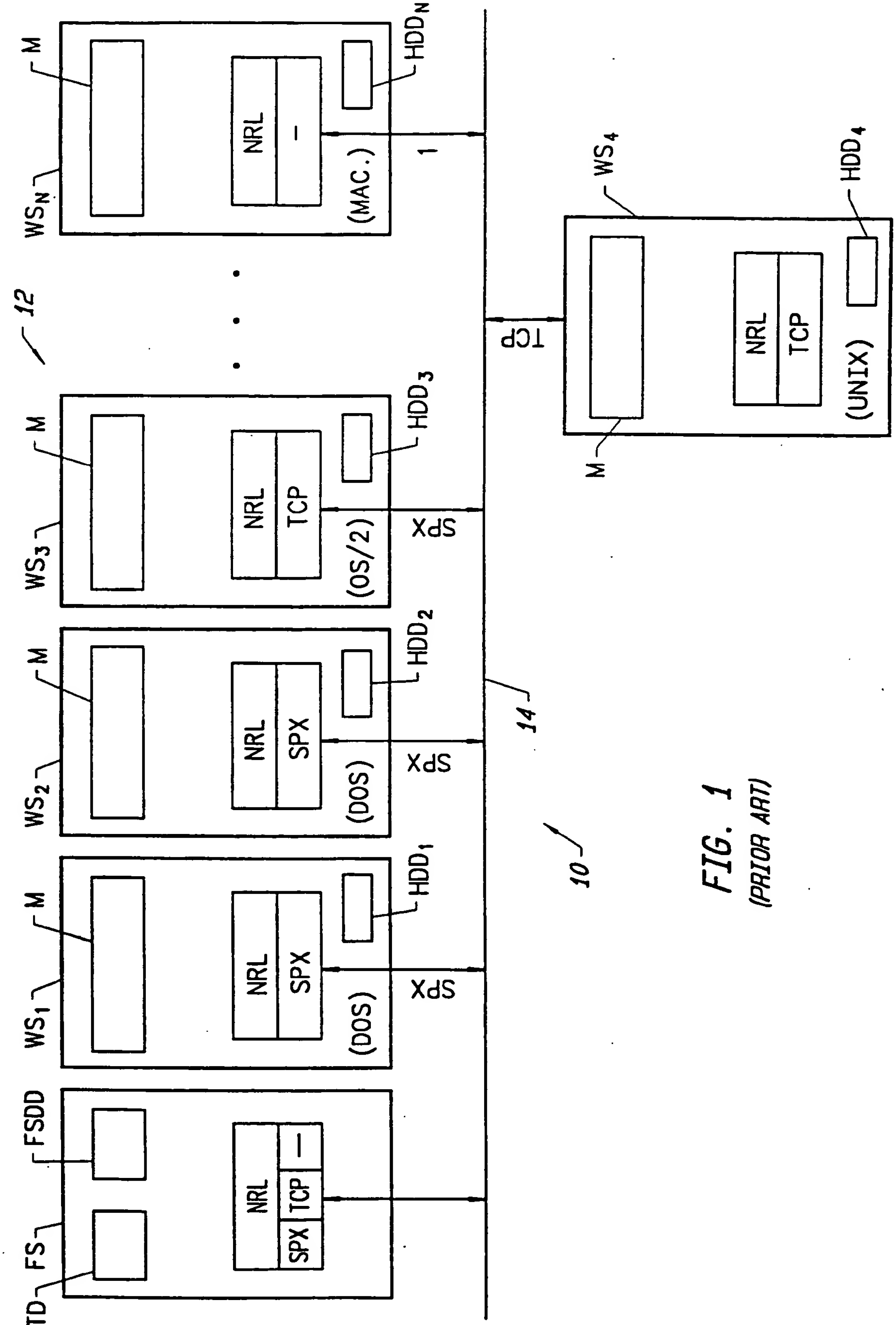
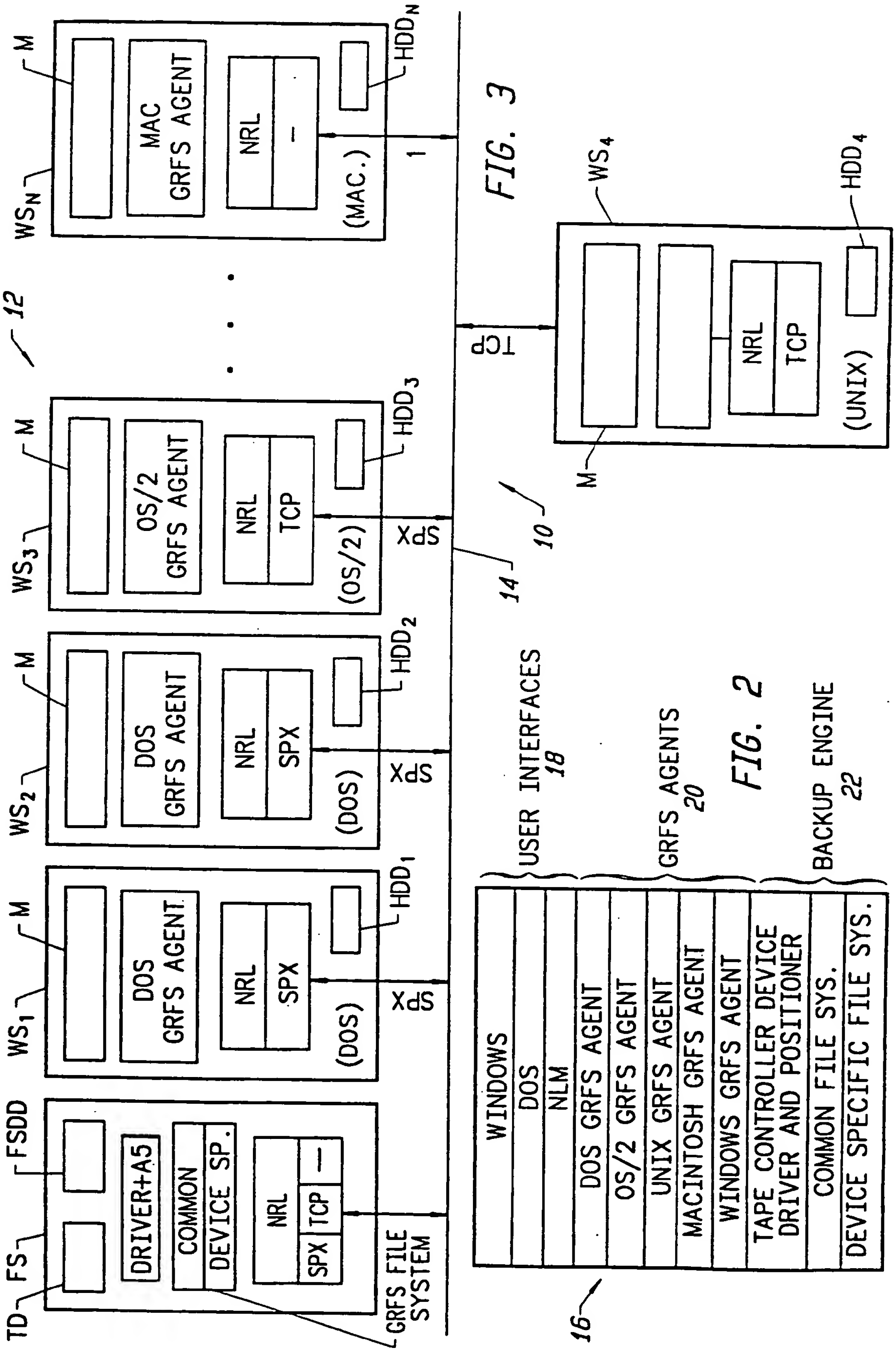


FIG. 1
(PRIOR ART)



TD
FS
FSDD
DRIVER+A5
COMMON DEVICE SP.
NRL
SPX
TCP
GRFS FILE SYSTEM
WS1
M
DOS GRFS AGENT
NRL
SPX
(DOS)
HDD1
WS2
DOS GRFS AGENT
(DOS)
HDD2
WS3
OS/2 GRFS AGENT
TCP
(OS/2)
HDD3
12
WSN
MAC GRFS AGENT
(MAC.)
HDDN
14
10
WS4
(UNIX)
HDD4
USER INTERFACES 18
GRFS AGENTS 20
BACKUP ENGINE 22
WINDOWS
DOS
NLM
DOS GRFS AGENT
OS/2 GRFS AGENT
UNIX GRFS AGENT
MACINTOSH GRFS AGENT
WINDOWS GRFS AGENT
TAPE CONTROLLER DEVICE DRIVER AND POSITIONER
COMMON FILE SYS.
DEVICE SPECIFIC FILE SYS.
16

FIG. 3

FIG. 2

EXAMPLE 1: To backup a single 5000 byte file. MESSAGE FLOW DIAGRAM FOR BACKUP

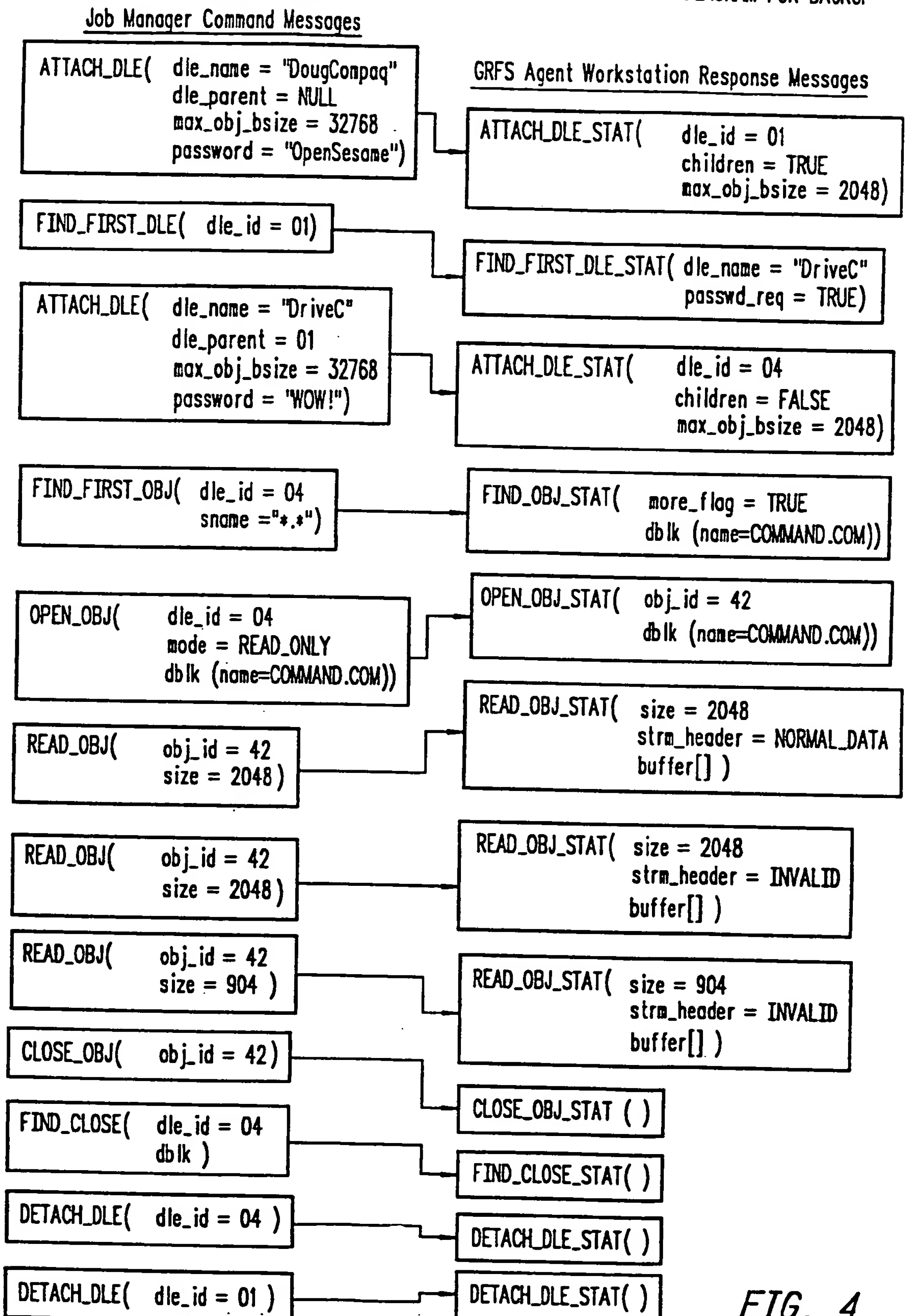


FIG. 4

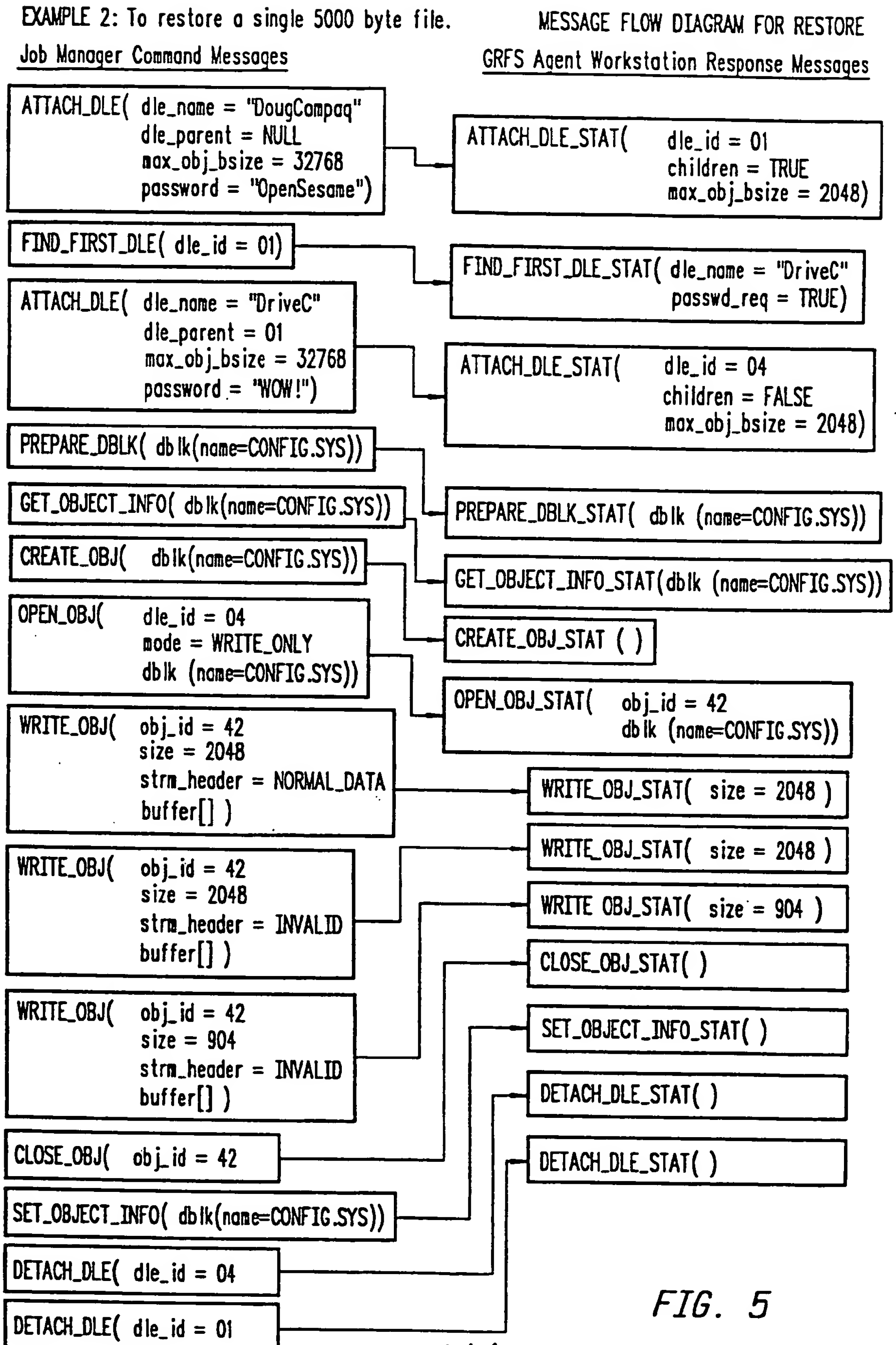

EXAMPLE 2: To restore a single 5000 byte file.
MESSAGE FLOW DIAGRAM FOR RESTORE
Job Manager Command Messages
GRFS Agent Workstation Response Messages
ATTACH_DLE(dle_name = "DougCompaq" dle_parent = NULL max_obj_bsize = 32768 password = "OpenSesame")
ATTACH_DLE_STAT(dle_id = 01 children = TRUE max_obj_bsize = 2048)
FIND_FIRST_DLE(dle_id = 01)
FIND_FIRST_DLE_STAT(dle_name = "DriveC" passwd_req = TRUE)
ATTACH_DLE(dle_name = "DriveC" dle_parent = 01 max_obj_bsize = 32768 password = "WOW!")
ATTACH_DLE_STAT(dle_id = 04 children = FALSE max_obj_bsize = 2048)
PREPARE_DBLK(dblk(name=CONFIG.SYS))
PREPARE_DBLK_STAT(dblk (name=CONFIG.SYS))
GET_OBJECT_INFO(dblk(name=CONFIG.SYS))
GET_OBJECT_INFO_STAT(dblk (name=CONFIG.SYS))
CREATE_OBJ(dblk(name=CONFIG.SYS))
CREATE_OBJ_STAT ( )
OPEN_OBJ(dle_id = 04 mode = WRITE_ONLY dblk (name=CONFIG.SYS))
OPEN_OBJ_STAT(obj_id = 42 dblk (name=CONFIG.SYS))
WRITE_OBJ(obj_id = 42 size = 2048 strm_header = NORMAL_DATA buffer[])
WRITE_OBJ_STAT(size = 2048)
WRITE_OBJ(obj_id = 42 size = 2048 strm_header = INVALID buffer[])
WRITE_OBJ_STAT(size = 2048)
WRITE_OBJ(obj_id = 42 size = 904 strm_header = INVALID buffer[])
WRITE OBJ_STAT(size = 904)
CLOSE_OBJ(obj_id = 42
CLOSE_OBJ_STAT( )
SET_OBJECT_INFO(dblk(name=CONFIG.SYS))
SET_OBJECT_INFO_STAT( )
DETACH_DLE(dle_id = 04
DETACH_DLE_STAT( )
DETACH_DLE(dle_id = 01
DETACH_DLE_STAT( )

FIG. 5

INTERNATIONAL SEARCH REPORT

Intern al Application No
PCT/US 94/12915

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F11/14

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	IBM TECHNICAL DISCLOSURE BULLETIN, vol.35, no.3, August 1992, NEW YORK US pages 286 - 289 'Centralized and rapid backup/restore for Work LAN File Services/VM' see the whole document	1-8
A	US,A,5 005 122 (GRIFFIN ET AL.) 2 April 1991 see abstract	1
A	US,A,5 133 065 (CHEFFETZ ET AL.) 21 July 1992 see abstract	8

[ ] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

'A' document defining the general state of the art which is not considered to be of particular relevance
'E' earlier document but published on or after the international filing date
'L' document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
'O' document referring to an oral disclosure, use, exhibition or other means
'P' document published prior to the international filing date but later than the priority date claimed

'T' later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
'X' document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
'Y' document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
'&' document member of the same patent family

1

Date of the actual completion of the international search	Date of mailing of the international search report
9 March 1995	17.03.95

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 340-3016	Corremans, G

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US 94/12915

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US-A-5005122	02-04-91	NONE	
US-A-5133065	21-07-92	NONE	